Vzdálený přístup k monitorovacím a řídícím systémům pomocí mobilních telefonů sítě GSM s využitím programů v jazyce JAVA

Jan Hamrlíček

Bakalářská práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan HAMRLÍČEK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Vzdálený přístup k monitorovacím a řídicím systémům pomocí mobilních telefonů sítě GSM s využitím programů v jazyce JAVA**

Zásady pro vypracování:

1. Zpracujte literární rešerši na dané téma.
2. Navrhněte a zrealizujte vzdálený přístup k monitorovacím a řídicím systémům pomocí mobilního telefonu sítě GSM s využitím datových přenosů GPRS.
3. Program pro vzdálený přístup v mobilním telefonu naprogramujte v jazyku JAVA pro mobilní telefony.
4. Vyhodnoťte dosažené výsledky s důrazem na kvalitu přenášených dat.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] PUŽMANOVÁ, R. Širokopásmový Internet – Přístupové a domácí sítě. Brno: Computer Press, 2004. ISBN 80-251-0139-8.
[2] HANUS, S. Bezdrátové a mobilní komunikace. Brno: VUT FEL, 2003. ISBN 80-214-1833-8.
[3] ŠMRHA, P., RUDOLF, V. Internetworking pomocí TCP/IP. České Budějovice: Kopp, 1994. ISBN 80-85828-09-X.
[4] GSM World – GPRS. Dostupný z URL: <http://www.gsmworld.com/technology/gprs/>

Vedoucí bakalářské práce: **Ing. Miroslav Matýsek, Ph.D.**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **14. února 2006**

Termín odevzdání bakalářské práce: **16. června 2006**

Ve Zlíně dne 14. února 2006

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této bakalářské práce je seznámení se současnými technologiemi přenosu dat v sítích GSM a technologií jazyka Java. Práce zároveň pojednává o jejich vzájemném využití při navržení a realizaci vzdáleného přístupu k monitorovacím a řídícím systémům. Tyto dvě oblasti zabírají největší část bakalářské práce. V praktické části jsou pak popsány vytvořené ukázkové programy v jazyce Java.

Klíčová slova:

Jazyk Java, GSM sítě, GPRS, J2SE, J2ME

ABSTRACT

Abstrakt ve světovém jazyce

The goal of this bachelor thesis is to intermediate a view of current technologies data transmission in GSM network and Java language technologies. The paper also deals with relative data assimilation in the course of proposition and implementation of remote access to monitoring and controlling systems. These two areas fill the greatest part of the bachelor thesis. In the practical part there are described created examples of programs in the Java language.

Keywords:

Java Language, GSM Network, GPRS, J2SE, J2ME

Rád bych chtěl poděkovat vedoucímu mojí bakalářské práce, kterým byl pan Ing. Miroslav Matýsek, za pomoc a rady, které mi poskytl a také za čas, který si pro mě našel během práce na mojí bakalářské práci. Také bych chtěl poděkovat za zapůjčení mobilního telefonu.

OBSAH

ÚVOD

Toto téma bakalářské práce jsou si vybral, protože mě zaujalo. Jsou v něm spojena dvě zajímavá témata a to programování v jazyce Java spolu s uplatněním vytvořených programů na mobilních telefonech. Díky tomu jsem se mohl seznámit s vlastním jazykem Java, který se v současné době těší velké oblibě a je vyžívám pro tvorbu aplikací na různých platformách a pro různé druhy zařízení. Zároveň jsem se seznámil s tím co obnáší vývoj programů pro mobilní telefony a co tyto zařízení nabízejí a jaké jsou jejich možnosti uplatnění.

I. TEORETICKÁ ČÁST

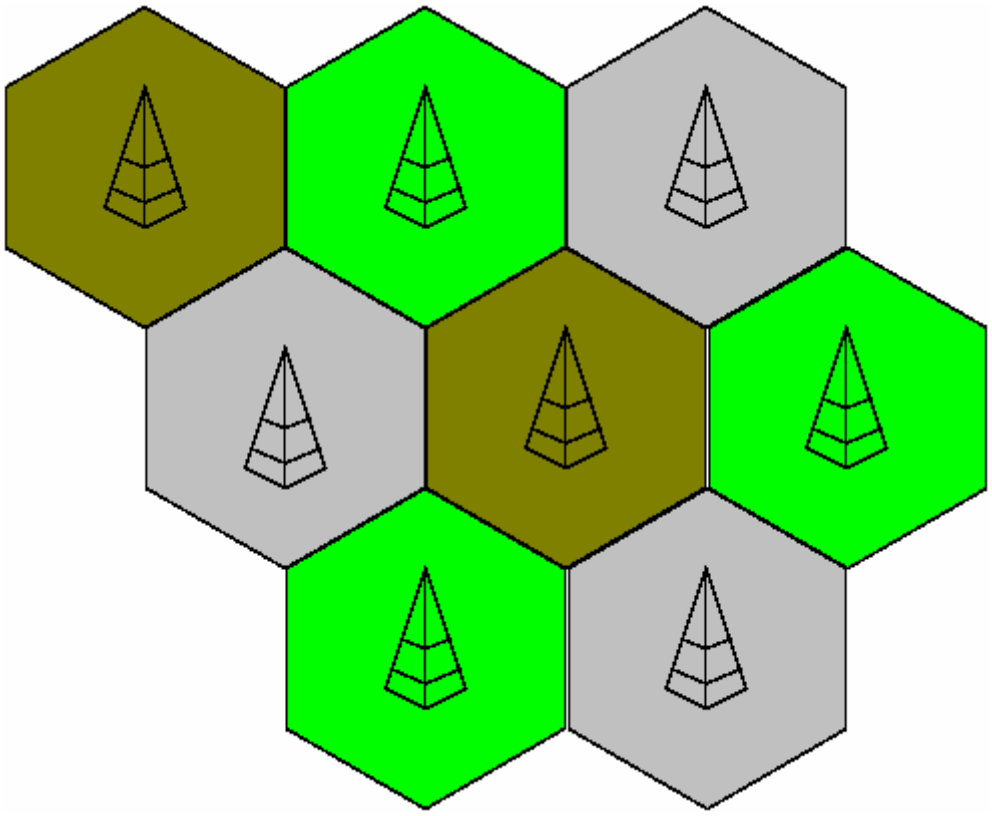
1 TECHNOLOGIE GSM A MOBILNÍ TELEFONY

Tato kapitola je seznámením s technologií GSM, která je využívána v současných mobilních bezdrátových sítích. Tyto sítě nás obklopují a využívá je prakticky denně miliony lidí například, když volají z mobilního telefonu. Zkratka GSM znamená **G**lobal **S**ystem for **M**obile Communications. GSM je druhá generace bezdrátových digitálních sítí v současnosti používaných na celém světě.

1.1 Princip sítě GSM

Mobilní sítě využívají ke svému fungování radiové vlny. U radiových vln je jedna ze základních charakteristik frekvence na které pracují. Aby se různé sítě vzájemně nerušily, musí být zaveden určitý systém, který říká, které frekvence může daná síť používat. Ovšem dostupných frekvencí není nekonečně mnoho a každý provozovatel sítě GSM získává jen omezený počet, který je mu přidělen v rámci licence, kterou získá od státního orgánu pověřeného správou frekvenčního spektra. V ČR je to Český telekomunikační úřad.

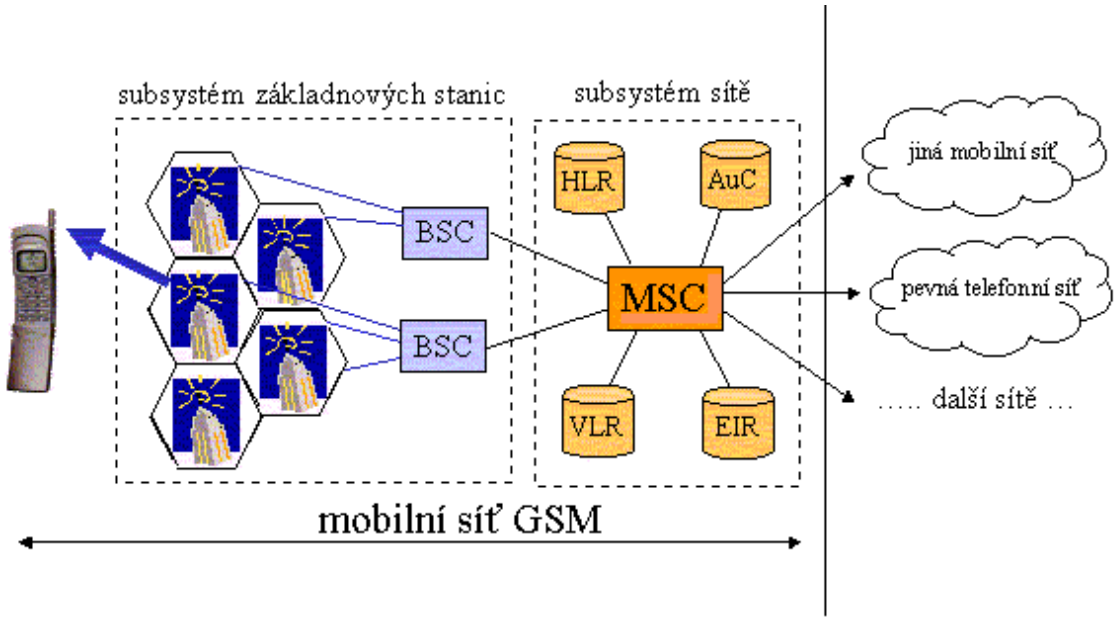
Protože přidělených frekvencí je jen omezené množství musel být zaveden určitý princip, na kterém budou GSM sítě fungovat tak, aby bylo možno obsloužit co nejvíc uživatelů sítě a zároveň pokrýt danou oblast tak, aby nevznikalo rušení. Proto jsou sítě stavěny na takzvaném celulárním (česky buňkovém) principu. Princip je takový, že daná oblast, která má být pokryta je rozdělena na vhodně velké oblasti – buňky. Tyto buňky jsou uspořádané tak, že když v určité buňce jsou používány určité frekvence z pásma, které má operátor přidělené, v žádné bezprostředně sousedící buňce nejsou používány tyto stejné frekvence a buňka používá jiné frekvence. V praxi se používá uspořádání se šestihrannými buňkami. Při tomto uspořádání lze vystačit již se třemi různými buňkami, v každé této buňce se používá jiný rozsah frekvencí a pomocí těchto buněk je možné pomocí jejich opakování pokrýt libovolně velké území. Počet frekvencí, které má buňka k dispozici také ovlivňuje, kolik například souběžných hovorů je možno přes danou buňku vést. Takže pokud chceme, aby bylo možné vést například více hovorů nebo datových přenosů, je nutné rozdělit danou oblast na více buněk.



Obrázek 1 Struktura sítě na buňkovém principu

Každá buňka obsahuje ve svém středu základovou stanici označovanou také zkratkou BTS (Base Transceiver Station). Každé koncové zařízení pak komunikuje pouze stou stanicí, která přísluší dané buňce. Komunikace vždy probíhá přes základovou stanici, ať už se cíl komunikace (například kamarád s mobilním telefonem) nachází ve stejné buňce nebo v jiné a tato komunikace probíhá na frekvencích, které má daná buňka přiděleny. Pokud se koncové zařízení pohybuje a přejde z jedné buňky do druhé, síť to pozná a dojde k předání komunikace na novou BTSku, která se nachází v oblasti do, které se zařízení přemístilo. Toto předání je pro uživatele zcela transparentní a neměl by ho vůbec pocítit.

1.2 Struktura sítě GSM



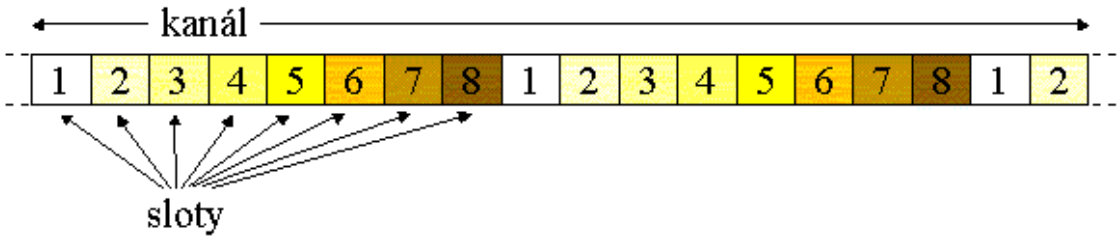
Obrázek 2 Struktura sítě GSM

Jednotlivé základové stanice jsou mezi sebou propojeny a společně řízeny. Obvykle několik základových stanic sdílí společnou řídící jednotku (BSC - Base Station Controller). Všechny základové stanice mobilní sítě jsou přes svoje řídící jednotky napojeny na centrální ústřednu (MSC - Mobile Services Switching Centre). MSC řídí sít a je také zodpovědné například za směrování hovorů k jejich příjemcům. Na MSC jsou napojeny ještě další pomocné systémy. V mobilní síti vzhledem k tomu, že koncová zařízení se mohou pohybovat musí být někde vedena evidence toho, kde se které zařízení momentálně nachází a právě k tomu složí registr HLR (HLR - Home Location Register). Dále někde musí být vedena evidence vlastních uživatelů a i návštěvníků sítě. K tomu slouží další registry napojené na MSC a to registr EIR (EIR - Equipment Identity Register), dále také AuC (AuC - Authentication Centre) a také VLR (VLR - Visitor Location Register). MSC je také napojeno do dalších sítí jiných mobilních operátorů a do pevné telefonní sítě.

1.3 Sdílení frekvencí

Jak už bylo uvedeno dříve, každá buňka má přiděleny určité frekvence, které používá. Těchto frekvencí - kanálů by však nemuselo být dostatečně mnoho, aby pokryly všechny požadavky, které se mohou vyskytnout. Proto se zavedla další technika, která umožní jednu frekvenci použít k přenosu více požadavků, například hovorů. GSM sítě proto používají

techniku časového multiplexu (TDM – Time Division Multiplexing, respektive TDMA - Time Division Multiple Access). Díky časovému multiplexu je možné jeden přenosový kanál rozdělit v čase na určitý počet stejných částí. To znamená, že klient komunikuje pouze určitou dobu, jejíž délka vyplývá z počtu částí na který byl přenosový kanál rozdělen a pak přepustí kanál dalšímu klientovi, který opět přepustí kanál dalšímu a tak se to opakuje až přijde řada opět na prvního klienta. Tento proces se pak neustále opakuje. Takže pokud je kanál rozdělen na 8 částí, může zároveň obsluhovat 8 klientů, kteří se v komunikaci pravidelně střídají vždy po určitém čase. Toto časové kvantum, které dostane klient přidělené se také označuje jako slot nebo se také používá termín timeslot.. Technologie GSM používá dělení na 8 částí, takže jeden přenosový kanál se rozdělí na 8 slotů a díky tomu je možné přes jeden kanál vést například 8 hovorů.



Obrázek 3 Rozdělení kanálu na sloty

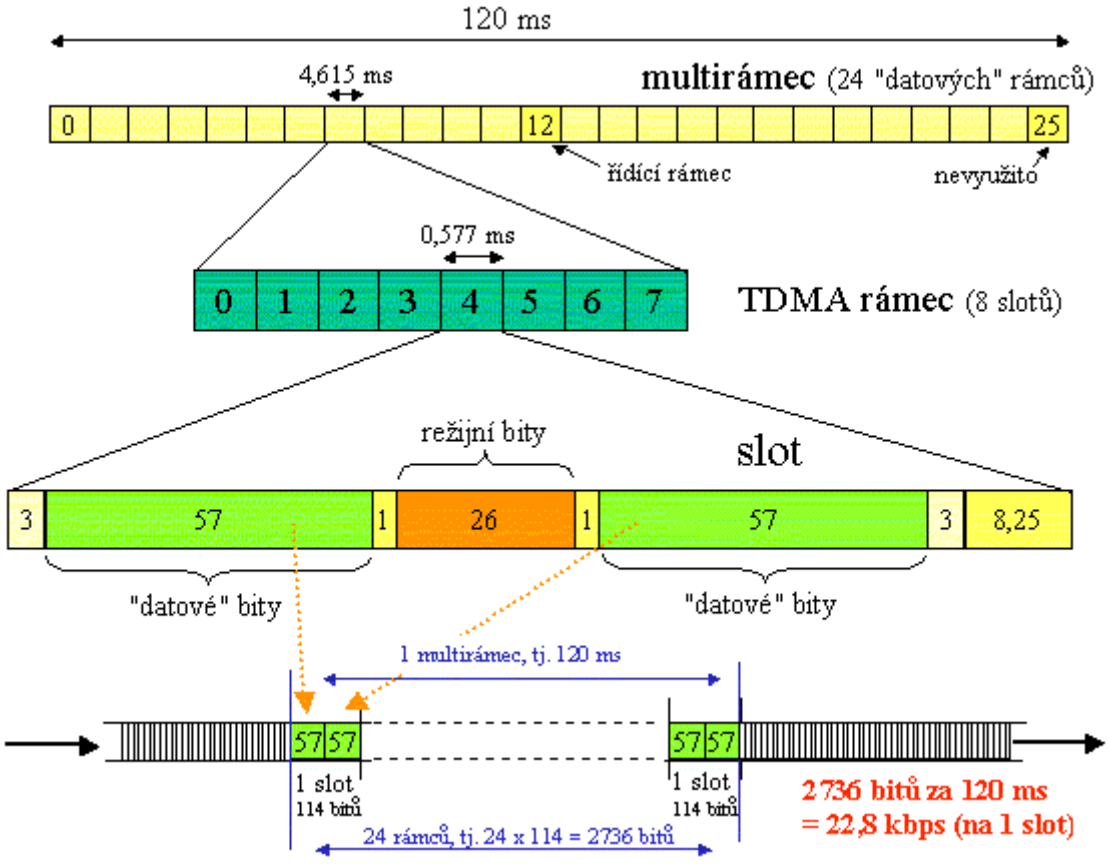
Díky technice časového multiplexu lze zjednodušeně říct, že počet hovorů vedených přes jeden kanál zvýší osmkrát.

V současnosti se v sítích GSM používají dvě základní pásma 900 MHz a 1800 MHz se šířkou kanálu, respektive odstupem kanálu 200 KHz. Tento samotný kanál by se teoreticky dal použít k přenosu dat, ale protože současné sítě byly navrhovány především pro přenos hlasu, pro který by byl kanál zbytečně velký, tak se jednotlivé kanály dělí pomocí techniky TDMA na 8 slotů. Z toho také pak vyplývá už dříve zmíněná schopnost obsloužit, například víc hovorů.

Jednotlivé sloty se pak po 8 seskupují a tvoří jeden TDMA rámec. Tyto rámce jsou pak seskupovány do takzvaného dopravního multirámce, který tvoří 26 TDMA rámců. Časová délka jednoho multirámce je 120 ms. Při přenosu je pak z multirámce pro vlastní přenos využito 24 rámců, 12 rámec je řídící a poslední rámec je vyhrazen. Jednotlivé mutirámce jsou při přenosu od sebe odděleny mezerou délky tří slotů. Časovou délku rámce tedy určíme jako časová délka multirámce (120 ms) děleno počtem rámců v multirámci (26 rám-

ců) a z toho nám vyjde časová délka jednoho rámce jako 4,615 ms. A délka rámce dělena počtem slotů v rámci nám určí časovou délku jednoho slotu, tedy 4,615 / 8 = 0,577 ms.

Důležitým aspektem také je vnitřní formát jednotlivých slotů, který ovlivňuje kolik dat jsme schopni přenést. Těchto formátů existuje několik druhů a nejčastější formát pak odpovídá datové délce 156,25 bitů na jeden slot. Necelý počet bitů je zapříčiněn tím, že na konci je speciální ukončovací posloupnost délky 8,25 bitů, která má zabránit, aby nedocházelo k překrývání signálu putujícího k základové stanici z různých terminálů. Ve vlastním slotu pak máme 2 x 57 bitů, které můžeme využít pro vlastní přenos. Zbytek jsou oddělovací a režijní bity. Díky této znalosti tedy můžeme určit kolik dat připadá na jeden slot v multirámci. To zjistíme tak, že vynásobíme 2 x 57 bitů a dostaneme počet užitečných bitů v jednom slotu a tento výsledek vynásobíme počtem datových rámců z multirámce. Nesmíme zapomenout, že daný slot se pravidelně opakuje v každém datovém rámci. Takže na jeden slot v mutirámci připadá celkem 2 x 57 x 24 = 2736 bitů.



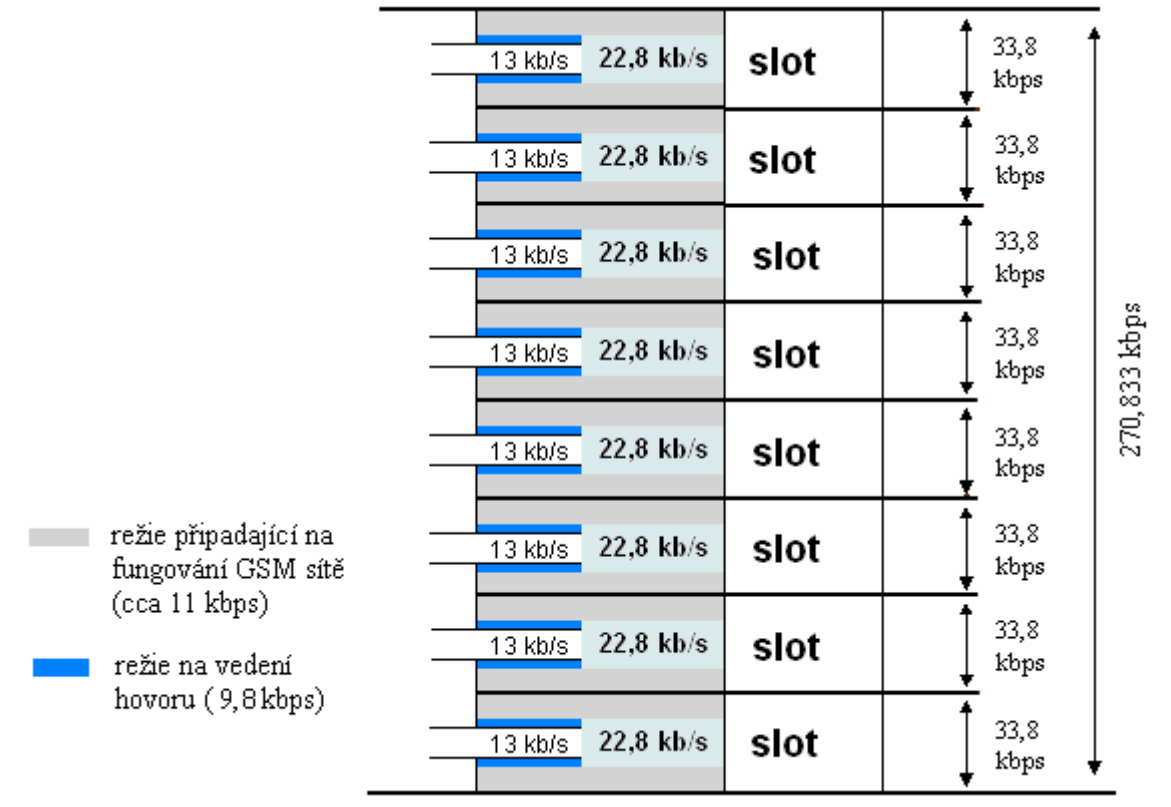
Obrázek 4 Znázornění struktury slotů, rámců a multirámců

Dále můžeme určit přenosovou kapacitu 1 slotu za sekundu. Tu určíme tak, že předchozí výsledek vydělíme 120 – což je počet milisekund potřebných k přenesení multirámce a

vynásobíme 1000 – což je počet milisekund v jedné sekundě a dostáváme tedy 2736 / 120 x 1000 = 22800 b/s. Na jeden slot tak připadá přenosová kapacita 22800 b/s. Těchto 22800 b/s je tedy využitelná přenosová kapacita bez potřebné režie nutné pro fungování GSM sítě. A tedy na celý přenosový kanál, který obsahuje 8 slotů připadá celkem 8 x 22800 = 182400 b/s = 182,4 kb/s využitelné přenosové kapacity. Pokud bychom chtěli zjistit teoretickou přenosovou kapacitu jednoho slotu za sekundu a z toho vyplývají velikost režie nutné k zajištění funkcí GSM sítě, tak to určíme jako 156,25 x 26 / 120 x 1000 = 33854 b/s. Teoretická přenosová kapacita jednoho slotu za sekundu tedy je 33854 b/s a z toho vyplývající celková teoretická přenosová rychlost celého kanálu je 270833 b/s ≈ 271 kb/s. Pokud tedy od teoretické přenosové kapacity jednoho slotu za sekundu odečteme využitelnou přenosovou kapacitu slotu za sekundu, tak dostaneme velikost režie nutné pro zajištění funkcí GSM sítě připadající na jeden slot. Tedy 33854 – 22800 = 11054 b/s ≈ 11 kb/s je velikost režie připadající na jeden slot.

1.4 Přenos dat

Jestliže chceme zjistit základní rychlost, kterou sítě GSM nabízí pro přenos dat, musíme vsít do úvahy původní určení sítí GSM pro přenos hlasu. Pro potřeby přenosu hlasu pomocí sítě GSM se musí hlas digitalizovat. Technologie GSM používá metodu, které snímá vzorek každých 20 ms a kóduje jej pomocí 260 bitů. Z toho nám vyplývá datový tok 13 kb/s, který je potřeba přenést. Tento datový tok se, ale musí nějak chránit proti různým chybám a zkreslením, které mohou nastat při přenosu. Proto se v GSM používají různá kódování pro různá přenášená data, které mají zajistit ochranu před nepříznivými vlivy. Pro přenos hlasu se používá konvoluční kódování, které k přenášeným datům přidává další data, která pak umožňují opravit některé chyby vzniklé při přenosu. Takže místo původního vzorku 260 pak dostáváme 456 bitů, které jsou stále generovány každých 20 ms. Za 120 ms což je délka jednoho multirámce tak dostáváme datový tok 456 x 6 = 2736 b/s což je přesně kapacita 1 slotu v multirámci. Z toho dále vyplývá celkový datový tok 22800 b/s, který je potřeba přenést a který přesně odpovídá přenosové kapacitě jednoho slotu za sekundu. Z tohoto celkového datového toku je tedy 13 kb/s užitečných dat a zbytek do 22,8 kb/s tedy 9,8 kb/s je režie na vedení hovoru.

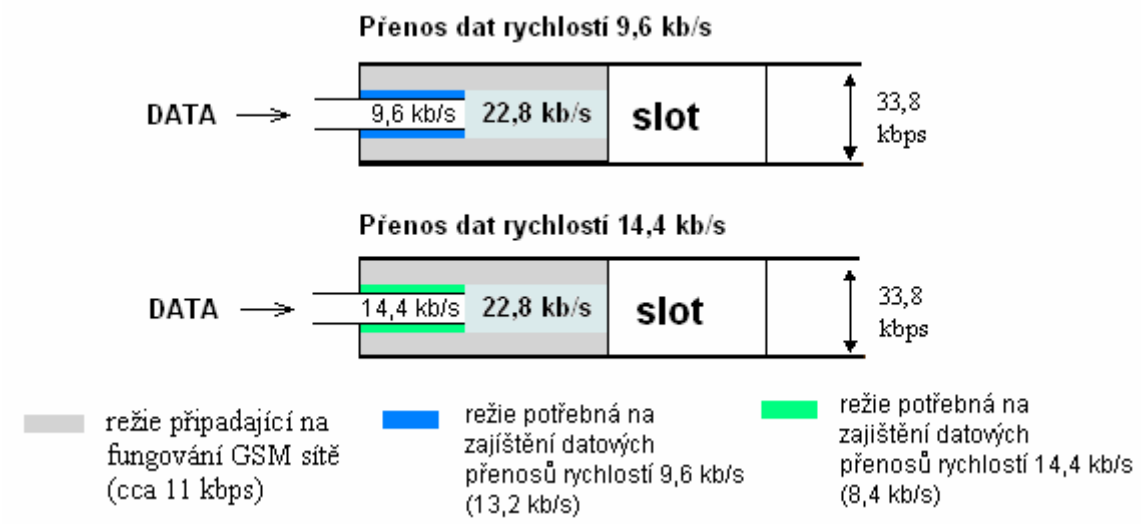


Obrázek 5 Struktura celého přenosového kanálu

1.5 CSD

Přestože byly GSM sítě navrženy hlavně pro přenos hlasu, později se objevily i požadavky na přenos dat. Teoreticky by bylo možno použít pro přenos dat přenosovou kapacitu jednoho slotu, tedy 22,8 kb/s, kterou získáme po odečtení režie na fungování sítě GSM. Ale ani těchto 22,8 kb/s se nedá použít, protože jak v případě hlasu tak i při přenosu dat je nutná určitá režie na zajištění přenosu. Pro data tedy byly původně zvoleny ještě robustnější techniky, které měly zajistit spolehlivost přenosu a díky tomu nám pak z celkových 22,8 kb/s zbylo pro vlastní data 9,6 kb/s a 13.2 kb/s tvoří režie pro zajištění přenosu. Rychlost 9,6 kb/s byla zvolena jako nejbližší nižší normovaná rychlost vzhledem k datovému toku, který vzniká digitalizací hlasu a to je už zmíněných 13 kb/s. Těchto 9,6 kb/s je základní přenosová kapacita, kterou nabízejí datové přenosy po sítích GSM označované jako CSD (Circuit Switched Data). S dalším vývoje se ukázalo, že datové přenosy nejsou až tolik citlivé na rušení a že je možné oslabit režijní složku a zvýšit datovou. Z původních 9,6 kb/s se tedy oslabením režijní složky, podařilo zvýšit datovou složku na 14,4 kb/s, ale s tím, že toto zvýšení vyžaduje kvalitnější signál. Písmena CS ve zkratce znamenají, že tato techni-

ka přenosu pracuje na principu přepojování okruhů, kdy mezi příjemcem a odesílatelem vzniká logická cesta s vyhrazenou přenosovou kapacitou, která je garantována po celou dobu spojení. Na druhou stranu, ale z pohledu síťových zdrojů to znamená, že tyto zdroje jsou vyhrazeny i když neprobíhá žádný přenos dat a jsou blokovány a nelze je přidělit dalším klientům. U tohoto typu přenosu dat se většinou platí za délku připojení a počet přenesených dat se nezohledňuje.



Obrázek 6 Přenos dat pomocí technologie CSD

1.6 HSCSD

Technologie HSCSD (High Speed Circuit Switched Data) byla navržena pro zajištění rychlejších datových přenosů při využití stávajících sítí GSM. Tato technologie tedy vychází z technologie CSD a také používá přenosovou rychlost 14,4 kb/s připadající na jeden slot. Rozdíl oproti CSD je, ale že technologie HSCSD umožňuje pro přenos dat využít více slotů zároveň a tím zvýšit přenosovou rychlost. Počet těchto slotů pak také udává maximální rychlost přenosu, kterou lze dosáhnout. Počet současně použitých slotů se může lišit pro směr k uživateli (příjem nebo download) a pro směr od uživatele (vysílání nebo upload), nebo může být pro oba směry stejný. Počet použitých slotů udávají jednotlivé třídy HSCSD. Těchto tříd je definováno tolik, aby pokryly celé spektrum přenosových rychlostí od minimální, kdy je použit pouze jeden slot jak pro vysílání tak i pro příjem, až do maximální při použití osmi slotů, které jsou použity v obou směrech.

Třídy HSCSD			
Třída	Maximální počet slotů		
	Vysílání	Příjem	Celkem
1	1	1	2
2	2	1	3
3	2	2	3
4	3	1	4
5	2	2	4
6	3	2	4
9	3	2	5
10	4	2	5
12	4	4	5
13	3	3	6
18	8	8	16

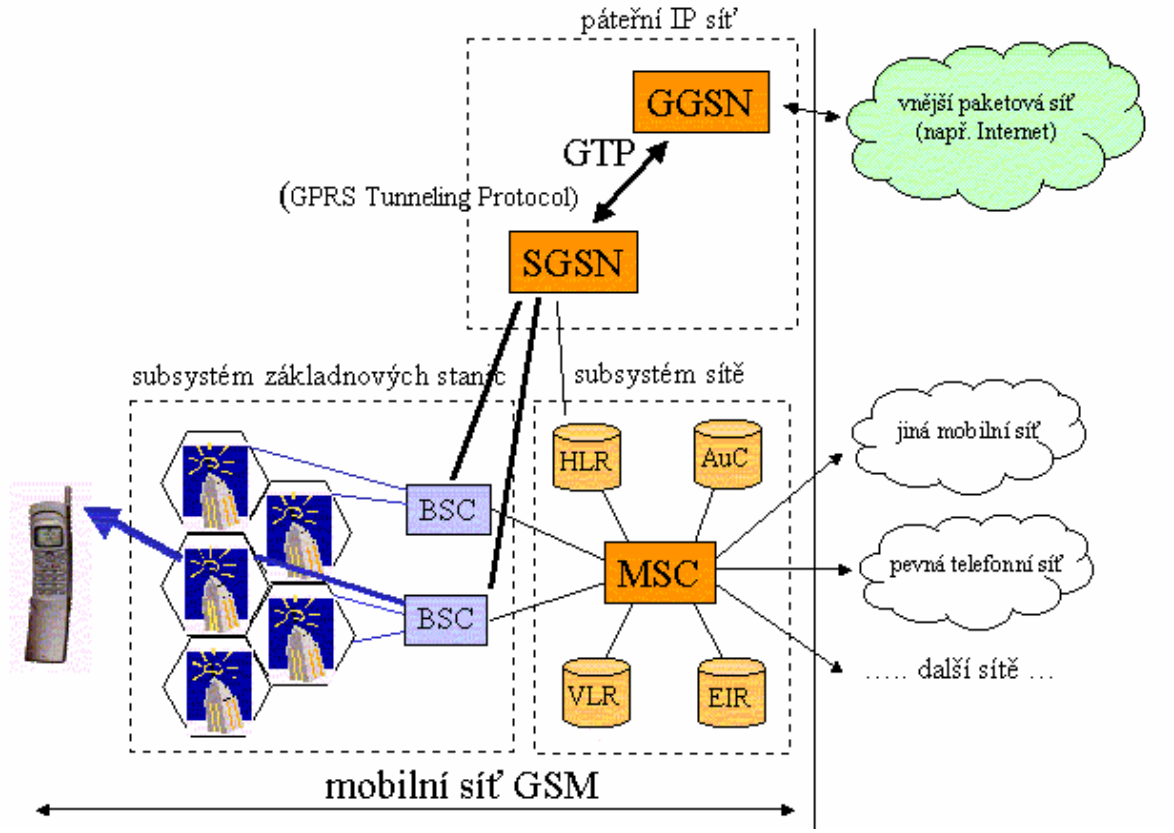
Tabulka 1 Třídy HSCSD

Pokud tedy chceme využít datové přenosy HSCSD, musíme mít telefon s podporou přenosů HSCSD a síť GSM určitého operátora musí podporovat přenosy HSCSD. To jsou základní požadavky, které musí být splněny. Pokud tedy chceme využít určitou třídu, rozhoduje o tom zdali dostaneme přidělený požadovaný počet slotů také to, jaké je momentální vytížení sítě. Jestliže například už probíhá větší počet hovorů, nemusí být požadovaný počet slotů k dispozici a my dostaneme přiděleno menší množství slotů, teoreticky by se mohlo stát, že pokud by daná základová stanice byla vytížena na 100%, tak už se na nás žádné volné sloty nedostanou. HSCSD také funguje na principu přepojování okruhů, to znamená, že je vyhrazeno požadované množství slotů a toto množství je k dispozici po celou dobu spojení. Tím máme garantovánu požadovanou rychlost přenosu dat po celou dobu. Takže připojení pomocí technologie HSCSD se hodí pro účely, kdy potřebujeme přenášet pravidelně data garantovanou rychlostí. U této služby se většinou platí za dobu spojení i s možným přihlédnutím k počtu použitých slotů.

1.7 GPRS

Technologie GPRS (General Packet Radio Service) je další technologie přenosu dat, které se dá v současných GSM sítích využít. Na rozdíl od předchozích technologií CSD a HSCSD však funguje na zcela jiném principu. GPRS funguje na principu přepojování paketů, tedy stejném principu jak například fungují počítačové sítě. Technologie GPRS tak jakoby vytvoří novou síť založenou na technice přepínání paketů přeloženou přes stávající

GSM síť. GPRS tedy využívá přenosové kapacity sítě jenom, když je potřeba odeslat nebo přijmout data, to znamená, že nejsou trvale vyhrazovány přenosové kapacity jako v případě CSD nebo HSCSD a není tak blokováno použití sítě dalšími uživateli. Díky technologii GPRS tak může využívat datové přenosy více klientů, než kdyby používali technologie fungující na bázi přepojování okruhů jako jsou CSD a HSCSD. Tento odlišný princip fungování na bázi přepojování paketů si ale také vyžádal určité úpravy sítě GSM.



Obrázek 7 Změny v síti GSM

Největší změnou je přidání dalších síťových prvků, které zajišťují podporu pro služby GPRS v sítích GSM. Jsou to uzly SGSN (Serving GPRS Support Node) a GGSN (Gateway GPRS Support Node).

Uzly GGSN plní úlohy brány mezi vnějšími datovými sítěmi, jako je třeba Internet a vlastní mobilní sítí podporující technologii GPRS. Tento uzel také ukládá informace nutné pro směřování příchozích paketů, které pak uzel SGSN posílá k jednotlivým klientům. Podporované vnější sítě připojené k uzlu GGSN mohou pracovat na bázi protokolu IP, nebo na bázi protokolu X.25.

Uzly SGSN doručují data do a z mobilních terminálů. Plní také další funkce související s vlastním fungováním služby GPRS, například připojování a odpojování mobilních terminálu a jejich ověřování. Uzly SGSN jsou přes řídící jednotky BSC napojeny přímo na jednotlivé základové stanice, kromě toho jsou napojeny také na některé registry z původní GSM sítě. Mimo jiné proto, aby bylo schopny zjistit kde se nachází jednotlivé mobilní terminály.

Standart GPRS také definuje tři třídy mobilních zařízení, které se liší schopností využívat zároveň služby na bázi přepojování paketů a přepojování okruhů.

- Třída A – zařízení třídy A mohou zároveň využívat služeb GPRS a také GSM služeb na bázi přepojování okruhů, což je přenos hlasu
- Třída B – zařízení třídy B mohou být v síti zaregistrované jak pro přenosy pomocí GPRS tak i pro GSM služby na bázi přepojování okruhů, ale v jednom okamžiku mohou používat pouze jednu službu
- Třída C – zařízení třídy C mohou být v síti zaregistrované buď pouze pro GPRS služby nebo pouze pro GSM služby. Přepínání mezi požadovanými službami se provádí manuálně

Technologie GPRS si také podobně jako technologie HSCSD, klade za cíl přenos dat vyššími rychlostmi než kolik nabízí pro přenos dat technologie CSD. K tomu GPRS používá možnost využít pro přenos více slotů a také tím, že zavádí nová kódovací schémata, která se snaží z přenosové kapacity jednoho slotu, která činí 22,8 kb/s, využít co nejvíc pro vlastní přenos dat a minimalizovat potřebnou režii.

Kódovací schéma	Přenosová rychlost [kb/s]
CS1	9,1
CS2	13,4
CS3	15,6
CS4	21,4

Tabulka 2 Používaná kódovací schémata technologií GPRS

Při použití kódovacího schémata CS4 využijeme skoro celou přenosovou kapacitu jednoho slotu pro vlastní data, ale také máme nejmenší schopnost opravy chyb vzniklých při přenosu. Naopak schéma CS1 má sice nejnižší přenosovou rychlost, ale zato je bude možné po-

užít i v podmínkách horšího signálu. Takže pokud použijeme pro přenos všech 8 slotů a kódování CS4 získáme maximální přenosovou rychlost, kterou mám může technologie GPRS nabídnout, tedy 8 x 21,4 = 171.2 kb/s. Konkrétní rychlost, kterou budeme moci při přenosu použít, závisí jednak na použitém mobilním přístroji (kolik slotů umí využít zároveň) a také na síti, kolik nám dovolí použít slotů a také na tom jaké používá kódovací schéma. Dostupnost služby GPRS také závisí na vytížení dané základové stanice, protože většinou mají hlasové přenosy nejvyšší prioritu, pak následují přenosy CSD a HSCSD a nejnižší prioritu mívají přenosy pomocí GPRS.

1.8 EDGE

Technologie EDGE (Enhanced Data rates for GSM Evolution) je poslední snahou o zvýšení přenosové rychlosti při použití stávající struktury sítí GSM. Tato technologie zachovává stávající frekvenční pásma a kanály, ale mění použitý způsob modulace. Původní dvoustupňovou modulaci používanou při vysílání rádiového signálu nahrazuje efektivnější osmistupňovou modulací. Díky tomu může teoreticky dosáhnout přenosové rychlosti až 384 kb/s při použití 8 slotů.

2 TECHNOLOGIE A JAZYK JAVA

Jazyk Java je v současnosti jeden z nejpopulárnějších programovacích jazyků, který je v současnosti používám pro hlavně pro programování aplikací na Internetu a také tam, kde je důležitá schopnost běhu aplikace na různých hardwarových a softwarových platformách. Vývoj samotného jazyka a souvisejících technologií je zajištěn v rámci takzvaného projektu Java Community Process (http://www.jcp.org). Tento projekt má zajistit, že vývoj bude probíhat po všeobecné dohodě všech zúčastněných. Na těchto webových stránkách má každý možnost se zapojit a ovlivnit tak vývoj jazyka a dalších technologií s ním spojených.

2.1 Vznik JAVY

Za vznikem jazyka Java stojí firma Sun Microsystems. Ta se zabývala problémem vývoje softwaru pro spotřební elektroniku. Brzy se zjistilo, že stávají programovací jazyky nejsou pro tyto účely nejvhodnější. Proto bylo rozhodnuto vytvořit nový programovací jazyk, který by bylo možné použít k vývoji softwaru pro různá zařízení spotřební elektroniky. Z toho důvodu byly stanoveny určité cíle a požadavky, které měly být při vývoji nového jazyka zohledněny a které také vyplývaly z původního určení pro spotřební elektroniku. Nový jazyk měl být nezávislý na architektuře, tak aby se dal použít na různých typech zařízení od elektronického diáře až po třeba ledničku. Dále měl být přenositelný, to znamená, že pokud fungoval na starém čipu měl by běžet i na novém. Jazyk měl být také spolehlivý, protože pokud se zařízení obsahují software napsaný v Javě pokazí vinou softwaru, výrobce většinou vyměňuje celé zařízení za nové, což sebou přináší zvýšené finanční náklady. Jazyk a software v něm napsaný měl být také robustní. To znamená, že by se měl umět vypořádat s různými nepředvídatelnými událostmi a stavy, které mohou nastat například v důsledku chyby uživatele zařízení. Nový jazyk měl být také jednoduchý, tak aby se ho každý zvládl rychle naučit a aby vývoj softwaru v něm byl rychlý a nebyl drahý. Vývoj jazyka také ovlivnil nástup Internetu. Díky těm původním požadavkům, které byly při vývoji jazyka zohledněny, si tvůrci jazyka uvědomili, že přesně splňuje požadavky, které jsou kladeny na software, který běží na počítačích připojených k Internetu. Především to byla jeho nezávislost na architektuře a přenositelnost. Rozvoj Internetu znamenal nový impuls pro vývoj Javy a v roce 1995 byla ohlášena první verze jazyka Java a také potřebných knihoven pro vývoj aplikaci. Zároveň také firma Sun představila jako demonstraci možností jazyka webový prohlížeč nazvaný HotJava, který byl napsán v tomto novém ja-

zyku. Podpora Javy byla také zabudována do tehdejších hlavních webových prohlížečů firem Microsoft a Netscape. Od této doby se začala Java rozšiřovat, firma Sun licencovala technologii Java dalším významným firmám, které ji pak implementovaly do svých produktů. Licence, ale také požadovala, aby firmy při implementaci Javy dodržely určité specifikace a požadavky, což mělo za následek, že program napsaný v Javě bylo možné spustit na různých implementacích Javy od různých výrobců. Od první verze Javy firma Sun pokračuje ve vývoji jazyka a pravidelně vydává nové verze, které opravují chyby obsažené v předešlých verzích a také přinášejí rozšíření například samotného jazyka a nebo dostupných knihoven potřebných pro vývoj, které se označují zkratkou API (Application Programming Interface). V současnosti (polovina roku 2006) je aktuální verze 5.0 nebo také označovaná jako 1.5.0. Od původního záměru požívat Javu pro psaní programů pro spotřební elektroniku se Java rozšířila i do oblasti počítačů a serverů, kde nachází široké uplatnění, dále je ji dnes možno použít i na různých typech přenosných přístrojů jako jsou například kapesní počítače a mobilní telefony. Na úspěch Javy reagovala firma Microsoft a uvedla podobnou technologii nazvanou .NET Framework. Toto jsou v dnešní době dvě hlavní technologie, na kterých probíhá vývoj a tvorba softwaru.

2.2 Vlastnosti Javy

Java je vyspělý moderní programovací jazyk, který obsahuje všechny důležité vlastnosti, které by měl moderní objektový jazyk obsahovat. Sama firma Sun ho charakterizovala jako jednoduchý, objektově orientovaný, distribuovaný, interpretovaný, robustní, bezpečný, nezávislý na architektuře, přenositelný, výkonný, podporující vlákna a dynamický jazyk. Tyto vlastnosti jsou pak popsány v následujících kapitolách.

2.2.1 Jednoduchost

Jedním z cílu vývoje bylo, aby byl nový jazyk jednoduchý a programátor se ho zvládl rychle naučit. Toho bylo dosaženo tak, že se za vzor nového jazyka vzaly populární jazyky C/C++, takže programátorům ovládajícím tyto jazyky nedělá problém přejít na Javu. Tvůrci jazyka se také snažili minimalizovat množství jazykových konstrukcí, které jazyk obsahuje. Z jazyka také byly odstraněny některé vlastnosti běžné v ostatních programovacích jazycích, které ale kód komplikují a jsou zdrojem chyb. Java tak nepoužívá hlavičkové soubory a preprocesor známý z jazyků C/C++. Dále Java neosahuje ukazatele a nepodpo-

ruje ukazatelovou aritmetiku, což je známý zdroj mnoha problémů, které se mohou vyskytnout při práci s pamětí. S tímto souvisí také to, že Java používá automatickou správu paměti, která se označuje GC (garbage collection). GC sleduje využívání paměti a alokované bloky paměti, na které již není žádný odkaz pak uvolní. Tím pádem se programátor nemusí starat o uvolňování alokované paměti.

2.2.2 Objektová orientace

Java byla od počátku navrhována jako čistě objektový jazyk. Základním stavebním kamenem v programu je třída, takže i ten nejjednodušší program musí obsahovat aspoň jednu třídu. Třída může obsahovat data a metody. Metody jsou obvykle určeny k tomu, že manipulují s datovými položkami třídy. Pomocí tříd je také možné lépe modelovat reálné problémy, které se řeší pomocí daného programu. Jedním z důležitých pojmů v objektovém programování je dědičnost. Pomocí dědičnosti je možné vytvářet nové datové typy (třídy), které získají vlastnosti a schopnosti objektu od kterého jsou odvozeny. Navíc v nové odvozené třídě je možné přidat další vlastnosti a nebo modifikovat zděděné vlastnosti. Dědičnost se uplatňuje například pokud máme výchozí základní třídu a od ní odvodíme další třídy, které pak představují specializovaný typ základní třídy.

Obrázek 8 Ukázka dědičnosti zobrazená pomocí UML

V Javě jsou všechny třídy odvozené od společného předka, ať už přímo či nepřímo. To zajišťuje, že všechny třídy sdílejí určité společné vlastnosti. Tohoto principu například využívá správce paměti (GC).

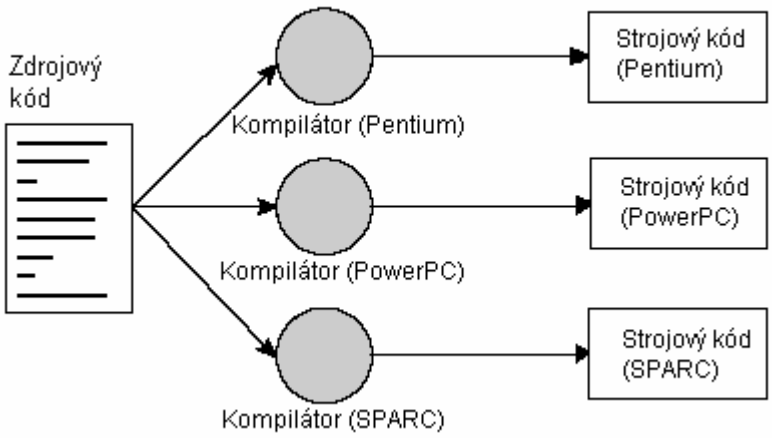
2.2.3 Distribuovanost

Java je navržena tak, aby podporovala vytváření aplikacích pracujících v sítích. Takže různé části aplikace můžou být různých místech v síti a všechno spolu bude spolupracovat.

Java také umožňuje používaní vzdálených zdrojů. Tato podpora sítí umožňuje v Javě celkem relativně snadno vytvářet různé síťové aplikace. Tyto aplikace můžou využívat různé druhy síťových spojení, které Java nabízí. Vytvoření síťové aplikace je Javě tak podstatně lehčí než ve většině ostatních jazyků.

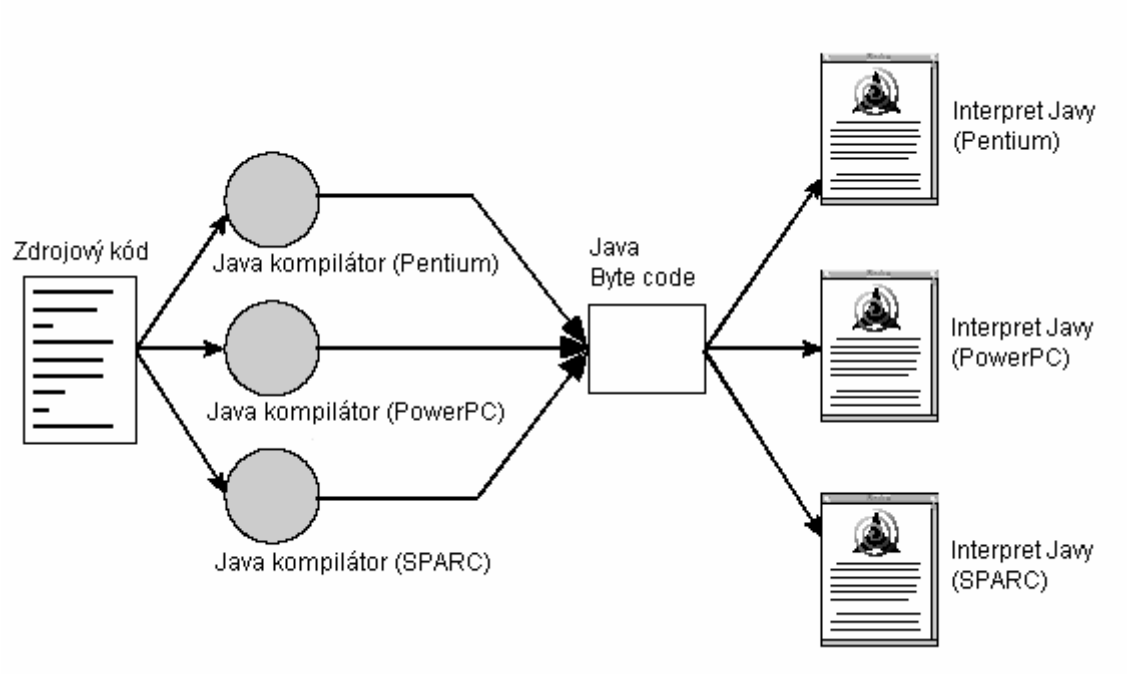
2.2.4 Interpretace

Kompilátor Javy nepřekládá zdrojové kódy přímo do strojového kódu určitého procesoru. Místo toho jsou zdrojové kódy přeloženy do bajtového kódu (byte code). Tento bajt kód je univerzální a je stejný pro všechny platformy, na kterých program poběží. Tento bajt kód je pak prováděn interpretem Javy. Interpret Javy je součástí takzvaného Virtuálního stroje Javy (Java Virtual Machine – JVM). JVM je nezbytný pro spuštění programu v Javě.



Obrázek 9 Překlad v tradičních kompilovaných jazycích

V tradičních kompilovaných jazycích je výsledkem kompilace soubor se strojovým kódem určitého procesoru. Proto se musí zdrojový kód přeložit pokaždé znovu pokud chceme, aby program běžel na nové platformě. To znamená, že musíme mít kompilátor, který přeloží zdrojový kód do strojového kódu daného procesoru. V Javě se zdrojový kód přeloží jen jednou do univerzálního bajtového kódu a tem je pak možné použít na všech platformách, na kterých je JVM. Díky tomu, že jsou programy v Javě interpretované jsou také pomalejší než programy přeložené přímo do strojového kódu daného procesoru.



Obrázek 10 Překlad programů v Javě

2.2.5 Robustnost

Program je robustní, pokud je spolehlivý a odolný proti různým chybovým stavům, které mohou nastat a je schopen tyto stavy nějak ošetřit. Java byla už od počátku navrhována tak, aby v ní šlo vytvářet robustní a spolehlivé aplikace, což taky bylo vynuceno původním určením pro tvorbu softwaru pro různé elektronické zařízení, ale co se ukázalo jako velmi vhodné i pro tvorbu softwaru na počítačích. V Javě se nejenom dá vytvářet robustní software, ale i sama Java je robustní. Při návrhu Javy tak byly vynechány některé prvky a vlastnosti běžné v ostatních jazycích, které jsou, ale zdrojem chyb. Java je silně typový jazyk, jsou vyžadovány explicitní deklarace proměnných a díky tomu je možné odhalit potenciální chyby už při komplici. Java také neobsahuje ukazatele, protože práce s ukazateli byla potencionálním zdrojem chyb. Další z důležitých vlastností Javy, která pomáhá snižovat možnost vzniku chyb při práci s pamětí je, že Java obsahuje automatického správce paměti (GC), který je součástí virtuálního stroje Javy. Takže programátor se stará pouze o alokovaní paměti. Uvolnění má pak na starost správce paměti, který se v nepravidelných intervalech spustí a alokované bloky paměti, na které již není žádný odkaz uvolní a vrátí do volné paměti. Velmi důležitou vlastností je, že Java podporuje výjimky a jejich strukturované ošetření. Kód, ve kterém muže dojít k nějakému nestandardnímu

stavu, při kterém může vzniknout výjimka, se vloží do takzvaného hlídaného bloku. Pokud výjimka vznikne, může být ošetřena v bloku obsluhy výjimky, který následuje hned za hlídaným blokem. Pokud výjimka není ošetřena postupuje směrem vzhůru a nakonec pokud není v programu ošetřena ji zachytí virtuální stroj Javy. Mechanizmus výjimek také přispívá k tvorbě přehlednějšího zdrojového kódu, který se lépe čte a udržuje.

2.2.6 Bezpečnost

Bezpečnost je jednou z důležitých vlastností Javy. Tato vlastnost je velmi důležitá zejména v prostředí Internetu. Java proto obsahuje bezpečnostní mechanismy, které se snaží snížit bezpečnostní riziko a které chrání uživatele proti nebezpečnému kódu. Tento kód uživatel může získat pokud se třeba dostane na webovou stránku, která obsahuje Java applet. Jedním z bezpečnostních prvků je model správy paměti v Javě. Díky tomu, že Java obsahuje automatickou správu paměti tak kompilátor nerozhoduje o umístění v paměti, takže dopředu nelze určit co kde bude uloženo a ani to nejde zjistit. Takže programátor se nemůže třeba pokusit narušit instanci třídy v paměti. Dále díky tomu, že Java neobsahuje ukazatele, tak je vyloučeno jejich zneužití, kdy by se třeba programátor pokusil například přesměrovat ukazatel na jinou část paměti a podstrčit tak škodlivý kód. Java také provádí verifikaci načítaných tříd, která zajistí, že daná třída má správnou vnitřní strukturu a pokud je nalezen problém je vyhozena výjimka. Bezpečnost v Javě je tedy zajištěna hlavně Virtuálním strojem Javy, který umožňuje omezit aplikaci v provádění určitých činností ke kterým nemá pověření.

2.2.7 Nezávislost na architektuře

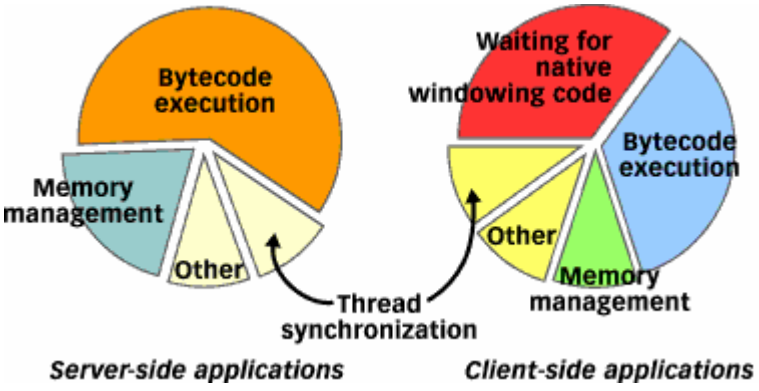
Nezávislost na architektuře je jedním z hlavních rysů Javy. Nezávislosti je dosaženo tím, že zdrojový kód se nekompiluje do strojového kódu určitého procesoru, ale do univerzálního bajtového kódu. Na rozdíl třeba od jazyků C/C++ kde výsledný spustitelný soubor je závislý jednak na typu procesoru pro který je překládám a typu operačního systému na kterém má program fungovat, výsledný spustitelný soubor vzniklý překladem zdrojových kódů v Javě je stejný pro všechny procesory a operační systémy na kterých má program běžet. Nicméně důležitou podmínkou je, aby pro daný procesor a operační systém byl implementován virtuální stroj Javy. Proto je Java vhodná pro tvorbu programů, kterými chceme oslovit co nejvíc uživatelů.

2.2.8 Přenositelnost

Nezávislost na architektuře je pouze částí, která umožňuje přenositelnost. Aby byla zaručena přenositelnost musí být také zamezeno existenci jakýchkoli implementačně závislých aspektů jazyka. Java tedy přesně specifikuje velikost primitivních datových typů a jejich chování při aritmetických operacích. Programy v Javě jsou tedy přenositelné a spustitelné všude tam kde je implementován virtuální stroj Javy. Proto musí být přenositelné i samotné běhové prostředí Javy. Tato přenositelnost se opírá o normu POSIX (Portable Operating System Interface).

2.2.9 Výkonnost

Rychlost provádění interpretovaných jazyků je většinou menší než u jazyků, které jsou kompilovány do strojového kódu. Java je interpretovaný jazyk, ale navzdory tomu je její rychlost provádění pro úlohy pro které se používá více než dostačující. Aplikace totiž většinou čekají na vstup od uživatele, nebo na data ze sítě či třeba databáze. Samotný výkon aplikace v Javě pak záleží především na celkovém návrhu aplikace, rychlosti s jakou je prováděn bajtový kód, rychlosti s jakou jsou prováděny knihovny (ve strojovém kódu) a také to ovlivňuje rychlost samotného hardwaru a operačního systému na kterém běží Virtuální stroj Javy.



Obrázek 11 Rozdělení celkového času běhu typické aplikace

Předchozí obrázek ukazuje, jak se liší provádění serverové a klientské aplikace a které části kolik zabírají z celkové doby běhu aplikace. Pokud je, ale potřeba větší rychlost, Java a virtuální stroj Javy obsahují některé další vlastnosti, které pomáhají zvýšit rychlost provádění programů v Javě. Jedním z nástrojů jak zvýšit rychlost je takzvaný Just-in-Time (JIT) překladač. JIT překladač je rychlý překladač, který překládá bajtový kód Javy do strojového kódu dané platformy na které běží za běhu aplikace. Metody jsou přeloženy JIT překla-

dačem těsně před jejich prvním spuštěním. Další z možností, kterou obsahuje virtuální stroj Javy jak zvýšit výkon aplikace je Adaptivní kompilace. Většina aplikací stráví velkou část svého běhu vykonáváním malé části kódu. Proto virtuální stroj Javy analyzuje aplikaci za běhu a identifikuje místa, která jsou výkonově kritická, ve kterých aplikace stráví nejvíc času vykonáváním bajtového kódu. Adaptivní kompilování tak místo aby kompilovalo celý program, když se spustí, nebo provádělo kompilaci celých metod jak to dělá JIT překladač, spustí program jako interpretovaný a analyzuje ho za běhu a hledá výkonově kritická místa. Pak pouze kompiluje a optimalizuje tyto kritická místa. Toto monitorování probíhá dynamicky po celou dobu běhu programu a automaticky se přizpůsobuje na základě chování aplikace. Tato metoda se nejvíc využije u aplikací běžících na serveru, protože ty jednak běží dostatečně dlouhou dobu aby optimalizace mohla proběhnout a také stráví nejvíc času vykonáváním bajtového kódu. Tyto a ještě další vlastnosti tak pomáhají zvýšit výkon aplikací v Javě. Ke zvyšování výkonu také obvykle dohází také s vydáním nové verze virtuálního stroje Javy. A nakonec ještě existují nástroje, které umožní přeloží bajtový kód přímo do spustitelné formy, která se používá na daném procesoru a operačním systému. Tím sice získáme rychlost jako mají tradiční kompilované jazyky, ale ztratíme možnost přenositelnosti programu.

2.2.10 Podpora vláken

Java obsahuje vestavěnou podporu vláken, se kterou se počítalo už při návrhu jazyka. To umožňuje používat Javu třeba pro vývoj aplikací, které musejí obsloužit naráz víc klientů. Právě pro tyto úlohy jsou vlákna vhodným řešením, kdy například webový server při přijetí požadavku vytvoří nové vlákno, které toho klienta obslouží. Ale vlákna se uplatní i jinde. Například jedno vlákno se stará o grafické uživatelské rozhraní, aby uživatel neměl pocit, že aplikace přestal reagovat, zatímco druhé provádí na pozadí nějaký náročnější výpočet. S vlákny souvisí také synchronizace, kterou také Java obsahuje jako součást samotného jazyka. Synchronizace se používá například proto, aby se zabránilo dvěma vláknům použít zároveň určitý prostředek. Pokud už jedno vlákno daný prostředek používá, pak pokus druhého vlákna o použití stejného prostředku způsobí, že dané vlákno se přeruší a musí čekat dokud první nedokončí práci a uvolní prostředek.

2.2.11 Dynamičnost

Java umožňuje dynamicky zavádět třídy do virtuálního stroje Javy. Tyto třídy mohou být uloženy třeba na disku a nebo mohou být zavaděny i ze sítě a to třeba až v případě, že jsou skutečně potřeba. Třídy v Javě také mají svoji jedinečnou reprezentaci v době kdy běží ve virtuálním stroji, je tak možné dynamicky zjistit ke které třídě objekt patří.

2.3 Edice Javy

Platforma Java má dnešní době široký záběr a neustále se rozrůstá. Rozsah použití Javy zahrnuje různorodá zařízení, které se ale mohou výrazně lišit svými schopnostmi a vlastnostmi. Proto se firma Sun rozhodla, že se celou platformu Java rozdělí do několika samostatných edicí. Tyto edice pak budou lépe přizpůsobené určitým zařízením na kterých se budou používat. Dále každá edice bude obsahovat to co je na daných systémech použitelné a využitelné, takže se hlavně bude lišit rozsah dodaných programovacích knihoven, která daná edice bude zahrnovat. Celá platforma Javy tak byly rozdělena do tří základních edicí. Tyto edice se jmenují Standart Edition, Enterprise Edition a Micro Edition.

2.3.1 Standart Edition

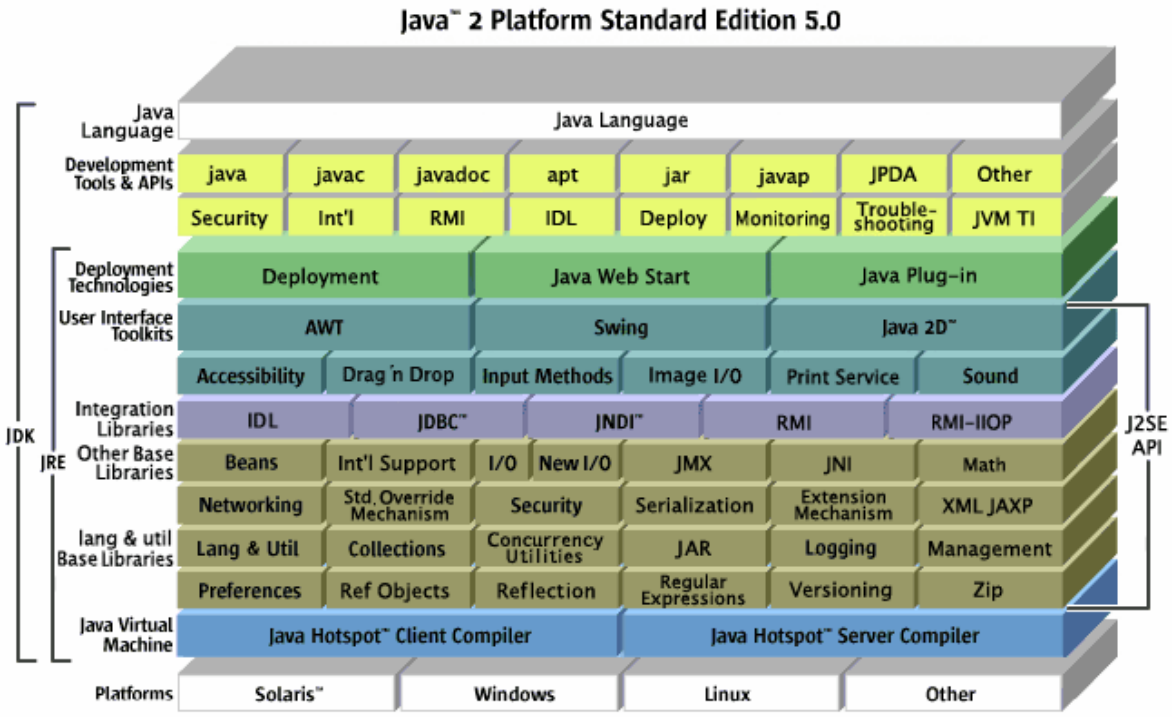
Tato edice bývá také označována jako J2SE (Java 2 Standart Edition) nebo Java SE. Toto je základní edice Javy, která nabízí kompletní prostředí vývoj a provozovaní aplikací na osobních počítačích a serverech. V současnosti se tato edice uplatňuje také v Embedded a Real-Time prostředích. Tato edice také tvoří základ Enterprise Edition. Hlavní použití této edice v současnosti je programování aplikací pro osobní počítače. Lze v ní vytvářet například aplikace s grafickým uživatelským rozhraním, nebo klientské aplikace, které přes síť komunikují se serverem. Pokud chceme programovat aplikace pro tuto edici musí být na počítači nainstalovaný J2SE Development Kit (JDK), který lze stáhnout z webových stránek společnosti Sun. Tento balík nástrojů obsahuje vše potřebné pro vývoj a ladění a testování aplikací v Javě. JDK tedy obsahuje například tyto části:

- Vývojové nástroje – sada nástrojů a utilit pomocí, kterých je možné vytvářet, spouštět, ladit a vytvářet dokumentaci k programům v Javě. Mezi nejdůležitější nástroje patří hlavně programy *javac* - to je kompilátor, *java* - je interpret programů v Javě, *javadoc* – je program, které ze zdrojových souborů vytvoří dokumentaci

v HTML formátu a program *appletviewer* – který umožňuje spouštět applety v Javě.

- Běhové prostředí (Runtime Environment) – obsahuje virtuální stroj Javy, potřebné knihovny a další soubory nutné pro běh programů napsaných v Javě
- Doplňkové knihovny – další knihovny a soubory důležité pro vývojové nástroje
- Ukázkové programy a applety – ukázkové programy a applety spolu s jejich zdrojovými kódy
- Hlavičkové soubory jazyka C – soubory, které umožňují používat Java Native Interface, což je rozhraní pomocí kterého mohou programy v Javě volat programy napsané v ostatních programovacích jazycích. Tak například můžeme použít knihovnu napsanou v jiném jazyce, která nemá v Javě alternativu v našem programu
- Zdrojové kódy – obsahuje zdrojové kódy základních tříd z Java API

Pokud chceme pouze spouštět aplikace stačí mít nainstalovaný J2SE Runtime Environment (JRE), který lze také stáhnout z webových stránek společnosti Sun. Celé JRE je jednou z částí JDK. Následující obrázek přehledně ukazuje jednotlivé standardní části Java 2 Standart Edition, tak jak je obsahuje implementace od firmy Sun.



Obrázek 12 Zobrazení jednotlivých částí Java SE

Jak je z obrázku patrné, rozsah Standart Edition je poměrně velký. Mimo jiné obsahuje například i dvě knihovny pro tvorbu grafických uživatelských rozhraní a to AWT (Abstract Window Toolkit) a Swing.

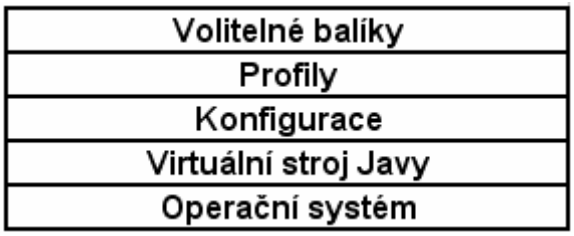
2.3.2 Enterprise Edition

Tato edice také bývá označována jako Java EE (Enterprise Edition). Java EE představuje standart pro vývoj více vrstvových, přenosných, robustních a bezpečných serverových Java aplikací. V této souvislosti se také používá termín Enterprise application, což jsou aplikace běžící na serveru, které jsou schopné poskytovat své služby zároveň velkému počtu klientů. Tyto aplikace jsou obvykle vytvořeny na míru pro danou firmu, a zajišťují firmě různé služby, které mohou například souviset s činností dané firmy. Java EE zjednodušuje vývoj aplikací díky standardizovanému systému komponent a služeb využívajících těchto komponent a také díky tomu, že implementační detaily chování aplikace jsou skryty a automatizovány bez potřeby složitého programování. Tato edice je založena na Standart Edition a k ní přidává další věci, které se uplatní především při programování serverových aplikací. To jsou například serverové komponenty Enterprise JavaBeans (EJB), které zapouzdřují obchodní logiku aplikace. Pokud chceme vytvářet Enterprise application je k tomu nutný

potřebný balík nástrojů, který je možno stáhnout z webových stránek společnosti Sun. Tento balík firma Sun označuje jako Java EE SDK (Software Development Kit).

2.3.3 Micro Edition

Tato edice také bývá označována jako J2ME (Java 2 Micro Edition) nebo Java ME. J2ME představuje ucelenou kolekci technologií a specifikací, která umožňuje vytvořit a provozovat aplikace v Javě na různých typech zařízení s omezenými zdroji. Těmito zdroji se myslí především výkon procesoru a kapacita dostupné paměti. Možná zařízení spadající do této edice jsou třeba pagery, mobilní telefony, různé set-top boxy až po třeba navigaci do auta. Protože rozsah zařízení, na kterých by tato edice Javy šla provozovat je velký a vzájemné rozdíly mezi jednotlivými zařízeními mohou být značné, například se mohou lišit výkonem procesoru, velikostí paměti, zobrazovacími schopnostmi, přítomností vstupních zařízení jako klávesnice nebo ukazovací pero a schopnostmi komunikace přes síť, tak došlo k rozdělení J2ME na menší části, které by pak lépe odpovídaly jednotlivým typům zařízení na které jsou určené. Java ME se proto dělí na konfigurace, profily a volitelné balíky.



Obrázek 13 Struktura Java ME

Základem na kterém běží Java ME je operační systém, který je provozován na daném zařízení. Může to být buď standardizovaný operační systém jako je například Symbian, nebo vlastní operační systém daného výrobce. Nad operačním systémem pak běží virtuální stroj Javy. Ten spolu s operačním systémem zajišťuje pracovní prostředí pro aplikace v Javě. Nad virtuálním strojem Javy jsou pak konfigurace. Konfigurace specifikuje vlastnosti použitého virtuálního stroje, podporované vlastnosti jazyka Java a také specifikuje základní sadu knihoven, která tak vytváří zkladní API pro vývoj aplikací pro určitou třídu zařízení. Konfigurace tak nabízí základní funkcionalitu společnou pro určitý rozsah zařízení, která je založená na společných vlastnostech těchto zařízení, jako je například velikost paměti, nebo možnost komunikace přes síť. Použitý typ virtuální stroje tedy závisí na dané konfiguraci a každá konfigurace má svůj vlastní virtuální stroj Javy. V současné době jsou defi-

novány dvě základní konfigurace a to CDC (Connected Device Configuration) a CLDC (Connected Limited Device Configuration). Tyto dvě konfigurace se vzájemně dosti odlišují.

Konfigurace CDC je určena pro zařízení, které mají výkonnější procesory a více paměti než zařízení pro konfiguraci CLDC. V současnosti je nejnovější verze konfigurace 1.1, která navazuje na předchozí verzi 1.0. Zařízení na kterých lze nalézt tuto konfiguraci jsou třeba navigační systémy do aut, nebo drahé mobilní telefony jako Sony Ericsson P910 nebo Sony Ericsson P990. Některé z požadavků, které udává specifikace CDC jsou:

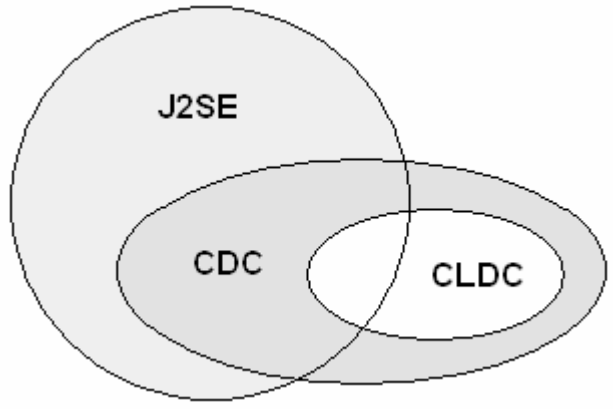
- Zařízení obsahuje 32-bitový procesor
- Zařízení má aspoň 2MB paměti pro Javu, zahrnuje jak paměť RAM i ROM
- Zařízení má připojení k nějakému druhu sítě, většinou bezdrátové
- Vyžaduje plně funkční virtuální stroj podle specifikace Java Virtual Machine Specification, 2nd edition

Současná verze CDC 1.1 vychází z J2SE 1.4.2 API, ze které přebírá docela velké množství tříd. Díky tomu umožňuje vytvářet propracované aplikace a programátoři zvyklí programovat pro J2SE mohou jednoduše přejít k programování pro J2ME/CDC. Protože tento profil nabízí poměrně široké možnosti musí také obsahovat odpovídající virtuální stroj, který bývá také nazýván CVM (Compact Virtual Machine). Tento virtuální stroj se podobá standardnímu virtuálnímu stroji z edice J2SE, který ale byl upraven, aby lépe odpovídal zařízením s omezenými zdroji, pro které je určený a dokázal využít jejich schopností a vlastností. Na této konfiguraci pak staví další profily, které přidávají další možnosti a jsou to profily Foundation, Personal Basis a profil Personal.

Druhá základní konfigurace se označuje CLDC. Tato konfigurace má mnohem nižší požadavky než konfigurace CDC a její snahou je vytvořit nejnižší použitelný společný základ pro J2ME zařízení.. V současnosti je tato konfigurace například hodně rozšířená v oblasti mobilních telefonů, kdy většina telefonů podporuje právě tuto konfiguraci a pouze pár telefonů podporuje konfiguraci CDC. Podobně jako konfigurace CDC i konfigurace CLDC prochází vývojem a v současné době je aktuální verze CLDC 1.1, která navazuje na verzi 1.0. Ve verzi 1.0 nebyly například podporovány datové typy s plovoucí desetinnou čárkou, které ale ve verzi 1.1 už podporovány jsou. Specifikace CLDC udává následující požadavky a vlastnosti, které mají mít zařízení určená pro konfiguraci CLCD:

- Zařízení má minimálně 192 KB paměti pro Javu, zahrnuje jak paměť ROM i RAM
- Zařízení obsahuje 16 bitový nebo 32 bitový procesor
- Zařízení má omezený zdroj energie, například bateriové napájení
- Zařízení lze připojit k nějakému druhu sítě, většinou bezdrátovým spojením s omezenou šířkou pásma

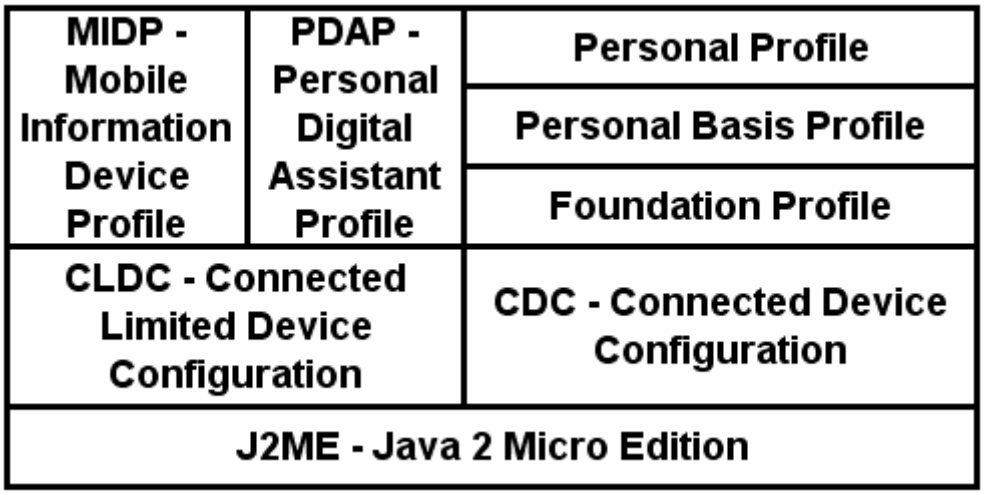
Díky tomu, že konfigurace CLDC je zamýšlena jako nejnižší společná platforma v prostředí J2ME, obsahuje proto jenom pár základních tříd z edice J2SE a k tomu přidává třídy specifické pro tuto konfiguraci. Nabízené API je tak podstatně chudší než u profilu CDC a vytvářené aplikace jsou zejména po stránce grafického uživatelského rozhraní velmi omezené. Tato konfigurace také obsahuje svůj vlastní virtuální stroj, který bývá také označován jako KVM (dřív zkratka znamenala Kilobyte Virtual Machine, dnes už znamená pouze K Virtual Machine). Tento virtuální stroj byl speciálně navržen pro tuto konfiguraci, tak aby přenositelný a měl nízké paměťové nároky. Z tohoto důvodu KVM postrádá některé vlastnosti běžné ve virtuálních strojích z konfigurace CDC a z virtuálního stroje používaného v J2SE. Například KVM nepodporuje verifikaci tříd tak jak ji provádí virtuální stroj v J2SE, protože tento proces by byl příliš náročný, ale verifikace je nyní rozdělena a část provádí překladač a KVM pak provádí pouze jednoduchou verifikaci. KVM také podporuje pouze základní sadu výjimek z edice J2SE. Na této konfiguraci pak staví profil MIDP (Mobile Information Device Profile) a profil PDAP (Personal Digital Assistant Profile).



Obrázek 14 Vztah konfigurací z J2ME k J2SE

Předchozí obrázek číslo 9 udává vztah jednotlivých konfigurací z J2ME ke standardní edici J2SE. Jak je vidět obě konfigurace obsahují větší či menší část z J2SE API, ale zároveň k tomu přidávají své vlastní specifické API, jako je například balík javax.microedition.io z konfigurace CLDC, který obsahuje takzvaný Generic Connection Framework, který má na starosti síťovou komunikaci. To mimo jiné zabraňuje přenesení aplikací napsaných zejména pro konfiguraci CLDC a používajících specifické API konfigurace CLDC do prostředí J2SE bez potřebných úprav.

Jak už bylo uvedeno dříve, nad konfiguracemi se nacházejí jednotlivé profily. Profily využívají základ, který poskytují jednotlivé konfigurace a k tomu přidávají další vlastnosti a možnosti, zejména rozšiřují dostupné API. V důsledku toho můžou profily také zvětšovat požadavky na daná zařízení. Jednotlivé profily také mohou zmenšovat množinu zařízení na kterých jdou použít oproti konfiguraci ze které vychází díky tomu, že se zaměřují pouze na určitý druh zařízení jako jsou například mobilní telefony. Díky tomu, ale mohou třeba nabídnout přístup k vlastnostem specifickým pro tyto zařízení. Jednotlivé profily také prochází vývojem a nová verze profilu většinou vychází z nové verze konfigurace, která tvoří základ profilu.



Obrázek 15 Struktura Java 2 Micro Edition

Profil Foundation vychází z konfigurace CDC a je určený jako základní profil pro další profily na něm postavené. Tento profil rozšiřuje možnosti API, které poskytuje konfigurace CDC. Přidává například další podporu pro síťovou komunikaci, ale také zároveň nespecifikuje žádné API pro tvorbu uživatelské rozhraní. V současné době je poslední verze 1.1.

Profil Personal Basis slouží jako základ pro profil Personal. Oproti Foundation profilu přidává podporu jednoduchého uživatelského rozhraní. Jeho použití je například u zařízení, které jsou schopny zobrazit zároveň pouze jedno okno uživatelského rozhraní. V současné době je poslední verze 1.1.

Profil Personal navazuje na předchozí profil Personal Basis a dále rozšiřuje dostupné API. Přidává další možnosti pro tvorbu uživatelských rozhraní a také například přidává podporu pro tvorbu a používání Java appletů. Poslední verze profilu je 1.1.

Profil PDAP (Personal Digital Assistant Profile) je postavený na konfiguraci CLDC a už z jeho názvu je patrné, pro které zařízení je určený. Tento profil se dá označit jako prostředník mezi profilem MIDP z konfigurace CLDC a profilem Personal z konfigurace CDC. Používá totiž některé prvky a vlastnosti z profilu MIDP, jako je například aplikační model a zároveň používá i některé věci z profilu Personal, jako je například API pro tvorbu grafického uživatelského rozhraní, které je založené na knihovně AWT. Tento profil ale není moc rozšířený.

V současné době je velmi rozšířený a používaný profil MIDP (Mobile Information Device Profile), který také staví na konfiguraci CLDC. Jak z názvu vyplývá je určený pro mobilní informační zařízení, což jsou například mobilní telefony, na kterých je v současnosti velmi rozšířen a umožňuje vytvářet programy v Javě pro tyto zařízení a proto také byl použit při tvorbě programu pro mobilní telefon v této práci. Tento profil je v současné době ve verzi MIDP 2.0, která navazuje na předchozí verzi MIDP1.0. Profil MIDP 2.0 je zpětně kompatibilní a aplikace napsané pro profil MIDP 1.0 lze spouštět i v prostředí MIDP 2.0. Těmto aplikacím se také říká midlety podle základní třídy od které musejí být odvozeny. Profil MIDP předpokládá, že zařízení na kterém bude provozován má tyto následující minimální vlastnosti:

- Rozlišení displeje 96x54
- Barevná hloubka displeje 1 bit
- Poměr stran pixelů displeje 1:1
- Jedno nebo více vstupních zařízení z následujícího seznamu: jedno-ruční klávesnice, obou-ruční klávesnice nebo dotykový displej

- Navíc aspoň 256 KB trvalé paměti k požadavkům stanoveným konfigurací CLDC pro uložení implementace profilu MIDP
- 8 KB paměti pro data aplikací, které zůstanou zachována i při vypnutí zařízení
- 128 KB paměti RAM pro běh virtuálního stroje
- Obousměrné bezdrátové připojení
- Schopnost přehrávat zvuky a tóny, buď pomocí hardwaru a softwarového algoritmu

Dále profil MIDP také definuje některé další vlastnosti a požadavky jako třeba:

- Způsob dodání a platby za aplikaci
- Životní cyklus aplikace – definuje strukturu aplikace a způsob jejího řízení
- Elektronické podepisování aplikací a bezpečnostní model
- Zabezpečené transakce pomocí protokolu http
- Uživatelské rozhraní a události
- Ukládání dat v zařízení

Profil MIDP je tedy možné použít k vytváření aplikací v Javě pro malá přenosná zařízení jako jsou třeba mobilní telefony. Lze vytvářet například klientské síťové aplikace, které mohou využít, pro vytvoření grafického uživatelského rozhraní standardní komponenty, které profil nabízí a nebo lze vytvářet například hry díky tomu, že profil nabízí přímo zabudovanou podporu pro tvorbu her.

II. PRAKTICKÁ ČÁST

3 POUŽITÉ NÁSTROJE

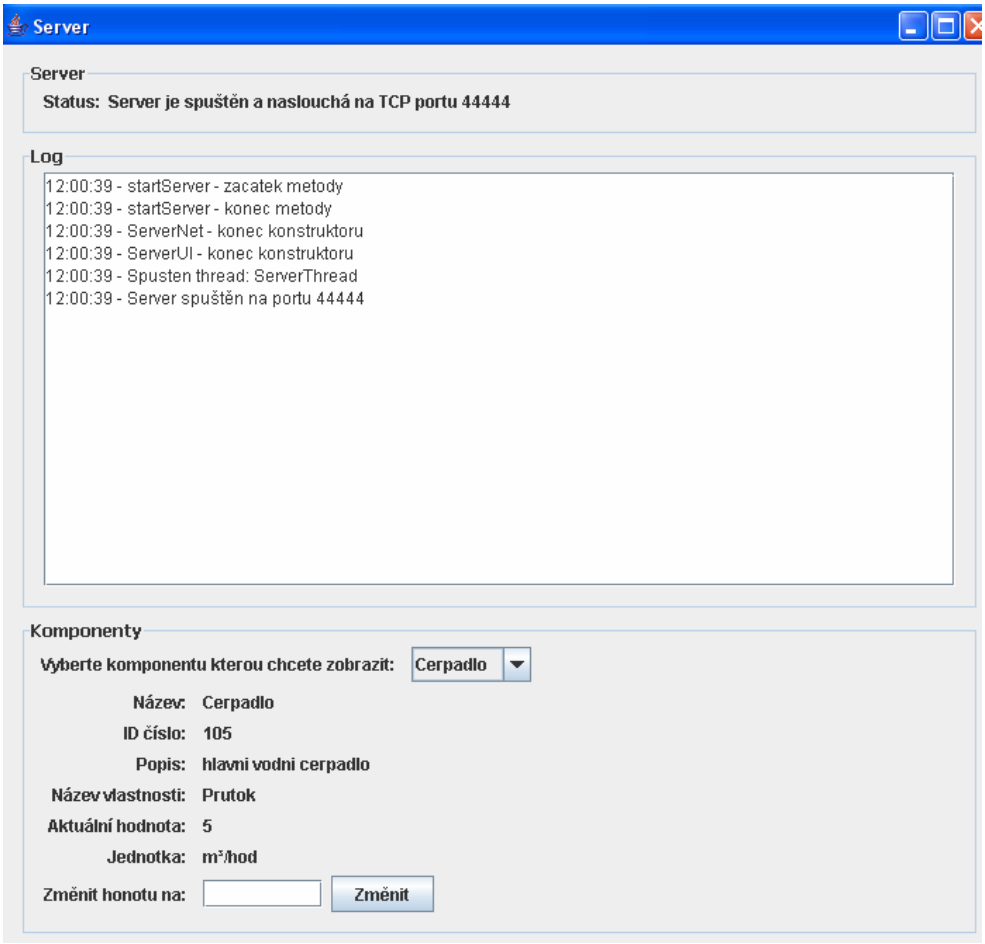
Abychom mohli vytvářet programy v Javě jak pro počítače, tak i pro mobilní telefony potřebujeme k tomu potřebné nástroje. Díky nim je pak vývoj rychlejší a také pohodlnější. Jako základ, který potřebujeme pro programování v Javě je mít nainstalovaný JDK (Java Development Kit). Jak už bylo zmíněno dříve, tento balík obsahuje potřebné nástroje pro vývoj programů v Javě. Pokud bychom chtěli pouze spouštět programy v Javě stačí mít nainstalované jenom běhové prostředí JRE (Java Runtime Environment). Tyto balíky lze stáhnout z webových stránek společnosti Sun. Dále je vhodné mít nainstalované nějaké vhodné vývojové prostředí, takzvané IDE. V našem případě připadaly v úvahu dvě vývojové prostředí a to Eclipse a NetBeans. Obě dvě podporují programování klasických aplikací v Javě a pro obě prostředí je i dostupná podpora pro vývoj v J2ME. Obě prostředí se také dají stáhnout zadarmo z Internetu. Nakonec bylo zvoleno prostředí NetBeans a to zejména z důvodu existence takzvaného Mobility Packu, což je doplněk do prostředí NetBeans, který toto prostředí rozšíří o podporu programování aplikací pro J2ME. Dále pak lze třeba do tohoto prostředí doplnit různé emulátory různých zařízení, zejména pak emulátory mobilních telefonů od jednotlivých výrobců.

3.1 Instalace použitých nástrojů

Jako první je třeba nainstalovat JDK. Lze ho stáhnout ze stránek společnosti Sun (http://www.sun.com). Stáhnout se dá buď samostatný JDK a nebo balíček, ve kterém je JDK společně s vývojovým prostředím NetBeans. Samostatné prostředí NetBeans lze pak stáhnout ze stránek (http://www.netbeans.org/). Pokud máme nainstalované běhové prostředí Javy a prostředí NetBeans, můžeme pak nainstalovat samotný Mobility Pack, který lze také stáhnout ze stejných stánek jako prostředí NetBeans. Dále byl ještě pro potřeby testování na emulátoru skutečného telefonu stáhnut vývojový balík pro telefony Nokia, konkrétně pro telefon Nokii 6230i. Tento balík se normálně nainstaluje, ale aby ho bylo používat v prostředí NetBeans, je nutné prostředí Netbeans pro jeho používání nastavit. Takže se spustí prostředí NetBeans, v menu Tools se vybere položka Java Platform Manager. V novém dialogu, který se objeví pak stiskneme tlačítko Add Platform a dále už postupujeme podle průvodce.

4 TESTOVACÍ PROGRAM PRO PC

Jelikož cílem této práce bylo vytvořit program v Javě pro mobilní telefon, který by se připojoval k nějakému serveru a sním komunikoval a získával od něho data a žádný vhodný univerzální server nebyl k dispozici, bylo potřeba vytvořit testovací server, který by odpovídal na požadavky klienta a jakoby simuloval chování nějakého řízeného systému. Tento testovací program je normální aplikace v Javě pro stolní počítače. Program používá jednoduché grafické rozhranní, k zobrazení informací o činnosti programu. Program byl vytvořen pomocí prostředí NetBeans, které má zabudovanou podporu pro vytváření grafických rozhraní programů, jako už prakticky všechny moderní vývojové nástroje. V tomto prostředí lze takovou aplikaci vytvořit pomocí přetahování komponent z palety komponent na formulář, kde pak jednotlivé komponenty rozmístíme. U těchto komponent lze také měnit některé z jejich vlastností přímo z vývojového prostředí. Základní zobrazení programu po startu je na následujícím obrázku.



Obrázek 16 Zobrazení testovacího programu

Program je rozdělen do tří částí. Úplně navrchu je pouze zobrazena informační hláška, že server, který naslouchá na příchozí spojení od klienta je spuštěný a na jakém portu naslouchá. V prostřední část je pak místo, do kterého se vypisují různé informace o tom co program dělá. Tyto informace se používaly hlavně jako pomoc při vývoji programu a také jako pomoc při odhalování různých chyb. V dolní části se pak nachází místo, ve kterém se zobrazují jednotlivé simulované systémy, které je možné jakoby ovládat a zjišťovat jejich parametry.

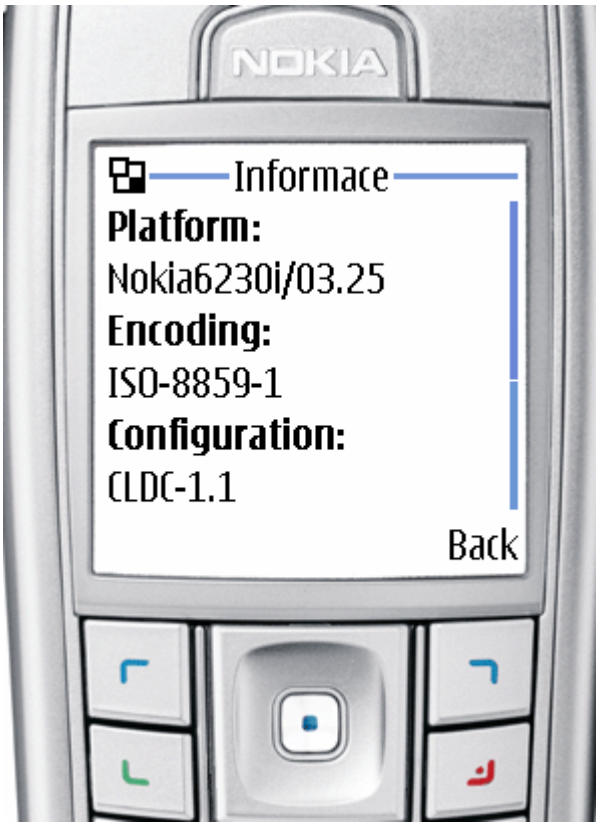
Vlastní zdrojové kódy programu jsou tvořeny několika třídami, které jsou všechny umístěny v balíku s názvem cz.utb.fai.javaserver. Třída, která má na starosti převážně grafickou stránku programu se nazývá ServerUI. V konstruktoru této třídy se vytváří všechny komponenty uživatelského rozhraní programu. Dále jsou v ní metody pro vypisování informací do grafického rozhraní a také kód v metodě main, který umožňuje změnit číslo portu podle parametru zadaného při spuštění programu, na kterém bude server očekávat spojení. Další důležitou třídou je třída ServerNet. Tato třída se vytváří v konstruktoru třídy ServerUI a přebírá jako parametr konstruktoru odkaz na hlavní okno aplikace, aby také byla schopna vypisovat informace na onrazovku. Tato třída má, ale hlavně na starost síťovou komunikaci. V této třídě tedy je metoda pro spuštění serveru na daném portu a další pomocné metody, které se využijí při komunikaci s klientem. Tato třída také obsahuje další dvě pomocné vnitřní třídy. A to třídu ServerThread a ClientThread, obě jsou odvozené od třídy Thread a díky tomu je možné je spustit jako samostatná výpočetní vlákny. Tento přístup byl zvolen, aby nedocházelo k zablokování uživatelské rozhraní síťovými funkcemi. V metodě run třídy ServerThread se vytvoří serverový socket, který naslouchá na určeném portu na příchozí spojení. Jakmile se připojí klient, dojde k vytvoření nového vlákna třídy ClientThread, které obstará vlastní komunikaci s klientem a jakmile se komunikace ukončí dojde i k ukončení tohoto vlákna. Server od klienta očekává příkazy v jenoducé formě podobné s formátem XML a klientovi vrací soubor ve formátu XML, který obsahuje požadované údaje. Další třída, které je použita v programu se jmenuje ServerLog, tato třída hlavně poskytuje metodu pro výpis různých hlášení do střední časti grafického rozhraní programu. Jinak neposkytuje žádné další podstatné funkce. A jako poslední třídu, kterou tento program obsahuje máme třídu ControlSystem, tato třída má za úkol jednoduchou simulaci jakoby reálných systémy, které by se daly ovládat. Tato třída tak obsahuje určité vlastnosti, které jakoby popisovaly reálný objekt. Obsahuje také určité pomocné metody, například pro výpis dat, které se pak odešlou klientovi.

5 PROGRAM PRO MOBILNÍ TELEFON

Cílem programu v Javě pro mobilní telefon bylo, aby byl schopen přes síť komunikovat s nějakým systémem. Díky tomu by pak šlo daný systém monitorovat a řídit na dálku. Díky tomu, že mobilní telefony jsou malé a snadno přenosné pak můžeme daný systém ovládat odkudkoli. Samotný program byl vytvářen ve vývojovém prostředí NetBeans spolu s nainstalovaným Mobility Packem a emulátorem mobilního telefonu Nokia. Díky Mobility Packu je možné vytvářet aplikace pro mobilní telefony podobným stylem jako u klasických programů. Mobility Pack také obsahuje paletu komponent a tyto komponenty a prvky se pak umísťují na plochu programu, kde se pak umísťují do takového pořadí jako budou mít v programu a definují se vztahy mezi nimi, zejména jednotlivé přechody mezi obrazovka programu v závislosti na vybraném příkazu. V nabídce komponent jsou tedy zejména různé obrazovky, příkazy které se na ně dají umístit, různé formulářové prvky a také různé nevizuální komponenty. Díky těmto komponentám je tak možné relativně snadno vytvořit různé aplikace, pro které bude jejich použití vyhovující. V našem programu jsem použili přístup takový, že část programu je vytvořena pomocí těchto komponent a zbytek byl dodělán ručně. Program používá jednoduché rozhraní tvořené pomocí textových menu a dalších prvků pro zobrazení a zadávání textu.

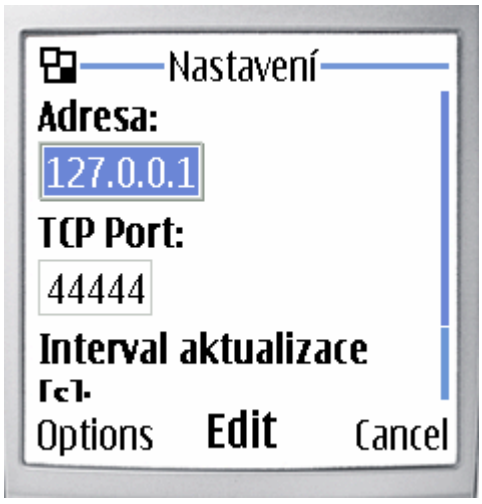
Obrázek 17 Hlavní nabídka programu

Jak je z předchozího obrázku patrné program na své první obrazovce obsahuje tři základní nabídky a taky možnost ukončení programu. Nabídka Informace zobrazí informace o zařízení na kterém program běží.



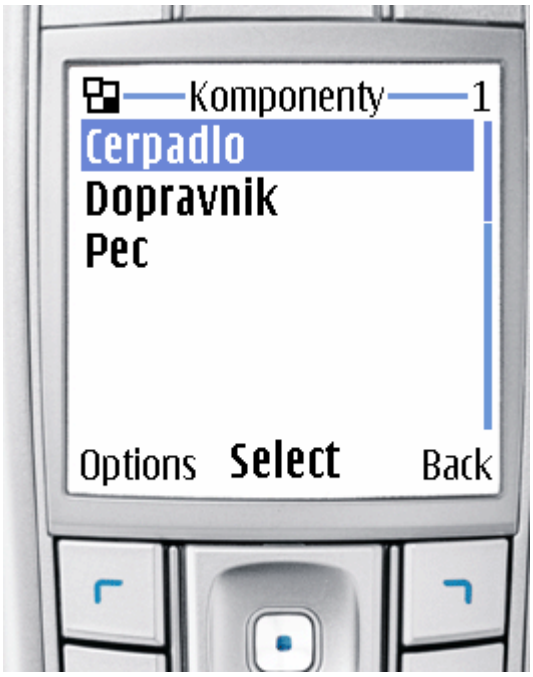
Obrázek 18 Zobrazení nabídky Informace v emulátoru telefonu Nokia

Druhá nabídka označená jako Nastavení umožňuje nastavit adresu a port serveru na který se má klient připojovat.



Obrázek 19 Zobrazení nabídky Nastavení

Po vybrání nabídky Připojit se telefon připojí k serveru pomocí údajů zadaných v nabídce Nastavení. Od severu pak získá data a zobrazí pak nabídku obsahující seznam simulovaných systémů, které je možné monitorovat a měnit jim parametry.



Obrázek 20 Zobrazení názvů simulovaných systémů nebo strojů

Po vybrání některé z položek, které jsou zobrazeny na předchozím obrázku se dostaneme na další obrazovku, která zobrazuje další informace o vybraném systému, nebo zařízení a umožní také nastavit novou hodnotu u vybrané vlastnosti ,která se pak zpětně odešle na server.

Obrázek 21 Zobrazení podrobností

Program pro mobilní telefon se také podobně jak testovací program pro PC skládá z několika tříd. Všechny tyto třídy jsou uloženy v balíku s názvem cz.utb.fai.j2meclient. Třída J2MEClientUI je třída, která má na starosti uživatelské rozhraní programu. Velká část této třídy je generována automaticky na základě toho jaké komponenty při tvorbě použijeme a jaké mezi nimi definujeme závislosti. Mimo to obsahuje metody, které vypisují získané údaje ze serveru na displej zeřízení. Další důležitou třídou je třída J2MEClientNet. Tato třída má na starosti síťovou část programu. Obsahuje metody pro vytvoření a zrušení připojení k serveru, pro odeslání požadavku, pro příjem odpovědi od serveru a také pro zpracování přijatých dat. Poslední třídou je pomocná třída Systems, která slouží k uchování informací o jednotlivých systémech obdržených od serveru. Proto nabízí pouze metody pro nastavování a vracení datových členů třídy.

ZÁVĚR

Cílem této práce bylo pokud možno navrhnout a zrealizovat vzdálený přístup pomocí mobilního telefonu využívajícího aplikaci napsanou v jazyce Java. Jak je patrné z této práce, tak tento úkol je za použití současných mobilních telefonů, podporujících provozování aplikací v Javě celkem relativně snadné splnit a umožňuje tak prakticky každému s potřebnými znalostmi takovou aplikaci vytvořit a používat. Díky mobilitě, kterou mobilní telefony nabízí a při využití přenosových technologií jako třeba GPRS, tak může být provozování takovýchto aplikací efektivní a také za přijatelnou cenu.

Velká část této práce byla věnována teoretickému základu v podobě současných mobilních sítí GSM a jejich schopnosti přenosu dat a druhá velká část byla věnována jazyku Java a technologií s ním spojených. Z tohoto teoretického přehledu také vyplývá proč je Java v současnosti tolik rozšířená a ve spojení s vhodnou technologií přenosu dat nabízí zajímavé využití.

Ohledně návrhu této aplikace byla snaha vytvořit pokud možno univerzální sytém, ale jak se později ukázalo, tato myšlenka se nepodařila zcela realizovat. Proto tuto práci lze spíše považovat pouze za naznačení možností, které jsou k dispozici.

SEZNAM POUŽITÉ LITERATURY

[1] PUŽMANOVÁ, R. Širokopásmový Internet – Přístupové a domácí sítě. Brno: Computer Press, 2004. ISBN 80-251-0139-8.

[2] HANUS, S. Bezdrátové a mobilní komunikace. Brno: VUT FEL, 2003. ISBN 80-214-1833-8

[3] ŠMRHA, P., RUDOLF, V. Internetworking pomocí TCP/IP. České Budějovice: Kopp, 1994. ISBN 80-85828-09-X.

[4] GSM World – GPRS. Dostupný z URL:<http://www.gsmworld.com/technology/gprs/>

[5] BRŮHA, L. Java Hotová řešení. Brno: Computer Press, 2003. ISBN 80-251-0072-3

[6] QUSAY H. MAHMOUD Naučte se Java 2 Micro Edirion. Praha: Grada Publishing, 2002. ISBN 80-247-0444-7

[7] Stránky pro vývojáře aplikací pro mobilní telefony Nokia: <http://www.forum.nokia.com>

[8] Stránky pro vývojáře aplikací pro mobilní telefony: <http://www.benq-siemens.com>

[9] Další elektronické zdroje na internetu

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

GSM	**G**lobal **S**ystem for **M**obile Communications
API	Application Programming Interface
GPRS	General Packet Radio Service
MIDP	Mobile Information Device Profile
CSD	Circuit Switched Data
HSCSD	High Speed Circuit Switched Data
BTS	Base Transceiver Station
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
EDGE	Enhanced Data rates for GSM Evolution
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standart Edition

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

Bez příloh.