Poglavlje 3.

Konstruisanje video-igre za mobilni telefon

ARHIVA ARKADNIH VIDEO-IGARA

Godine 1979. kompanija Namco je predstavila video-igru Galaxian. Video-igra Galaxian je bila prva velika pucačka video-igra čija radnja se odvijala u svemiru i bila je prvi naslednik video-igre Space Invaders. Video-igra Galaxian je prethodnik video-igre Galaga, koja je najverovatnije najuspešnija svemirska pucačka video-igra svih vremena. U video-igri Galaxian, kao i u većini video-igara ovog žanra, upravljate svemirskim brodom koji vodite duž donje ivice ekrana i pucate na vanzemaljce koji se pojavljuju na vrhu ekrana. Video-igra Galaxian zauzima posebno mesto u istoriji video-igara, jer je to prva arkadna video-igra u kojoj se koristio pun spektar RGB boja (RGB - Red Gree Blue (crvena, zelena, plava)) za prikazivanje svih delova video-igre.

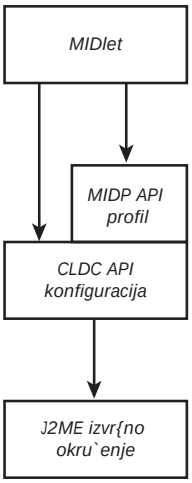
Java programiranje bilo koje vrste je vezano za programski jezik Java i skup API-ja koji podržava usluge aplikacija, kao što su GUI komponente, mrežna razmena podataka i ulaz/izlaz. Programiranje Java video-igara za mobilne uređaje se ne razlikuje jer se oslanja na skup API-ja koji podržava razne oblike funkcionalnosti koja je neophodna za MIDlet video-igre koje se moraju izvršavati u bežičnom mobilnom okruženju. Razumevanje ovih API-ja i njihovih mogućnosti je neophodno kako biste postali programer Java video-igara. U ovom poglavlju ću vam objasniti Java API-je za mobilne uređaje i pokazaću vam kako da napravite osnovu video-igre za mobilni telefon. Ova MIDlet osnova će nam poslužiti kao šablon koji ćemo koristiti za pravljenje video-igara u ovoj knjizi.

U ovom poglavlju ćete naučiti:

- Kako se J2ME programiranje može razvrstati u nekoliko različitih API-ja
- Internu strukturu MIDleta
- Kako da napravite MIDlet koji prikazuje važne informacije o mobilnom telefonu koje se mogu iskoristiti za video-igre
- Kako da za distribuiranje pripremite MIDlet

Istraživanje J2ME API-ja

Pre nego što se pozabavimo pisanjem programskog koda vaše prve video-igre za mobilne telefone, potrebno je da se ukratko upoznate sa API-jem koji se koristi za pravljenje MIDleta. MIDP (Mobile Information Device Profile) specifikacija je skup pravila kojima se opisuju mogućnosti i ograničenja standardnog skupa klasa i interfejsa programskog jezik Java koji se koristi za mobilne uređaje. Značajan aspekt mogućnosti i ograničenja jeste standardni skup klasa i interfejsa koji se mogu koristiti za MIDlet programiranje. Iako MIDP specifikacija sadrži detaljan opis API-ja koji možete koristiti za MIDlet programiranje, dodatni API-ji se mogu dobiti kroz CLDC (Connected Limited Device Configuration) konfiguraciju. MIDP API je izgrađen nad CLDC API-jem kako bi se obezbedile klase i interfejsi koji su specifični za mobilne uređaje. CLDC možete posmatrati kao nešto što obezbeđuje opšti Java API za mrežne uređaje, dok MIDP obezbeđuje detalljniji API na specifične osobine koje nisu deo CLDC API-ja za kompaktne bežične uređaje (kakvi su mobilni telefoni).



Slika 3.1
MIDlet mora pozivati CLDC i MIDP API kako bi obavio većinu funkcija.

Zbog čega biste bili zainteresovani za bilo koju od ovih specifikacija i API-ja? CLDC i MIDP specifikacije su važne jer eksplicitno definišu koje klase i interfejsi se mogu koristiti za pravljenje MIDleta. Mobilni uređaji su „spretne“ mašine koje nemaju megabajte memorije za smeštanje velikih aplikacija. Znajući to, kompanija Sun je morala da smisli način na koji bi omogućila korišćenje najmanjeg skupa funkcija, a da pri tom ne optereti previše mobilni uređaj. Njihovo rešenje je dvoslojni pristup koji se sastoji od konfiguracije i profila koji pruža detaljnije informacije. CLDC API opisuje osnovne klase i interfejse koji su neophodni za generički mrežni uređaj, dok MIDP API dodaje klase i interfejse koji su neophodni za mobilni uređaj (kakav je mobilni telefon). Na slici 3.1 vidite zavisnost između MIDleta i odgovarajućih CLDC i MIDP API-ja.

Zapamtite da, iako su CLDC i MIDP API-ji pažljivo osmišljeni kako bi bili što funkcionalniji u ograničenom memorijskom prostoru i sa ograničenim resursima mobilnih uređaja, oni ipak nisu dovolji u nekim situacijama. To znači da ćete ponekad, kao programer MIDlet video-igre, morati da se potrudite jer na raspolaganju nemate bogat API kakav se koristi za uobičajeno programiranje video-igara.

CLDC API

Većina klasa CLDC API-ja su direktno preuzete iz standardnog J2SE API-ja. Te klase i interfejsi su identični klasama i interfejsima koji se koriste u tradicionalnom Java programiranju. Ovaj deo CLDC API-ja je smešten u pakete koji imaju poznata J2SE imena, recimo, java.lang i java.util. Pored tih klasa i interfejsa, koji su direktno pozajmljeni iz J2SE API-ja, postoji nekoliko interfejsa koji su specifični za CLDC API. Ti interfejsi se uglavnom koriste za mrežnu razmenu podataka. To je oblast J2SE API-ja koju je naročito teško smanjiti za potrebe mrežnih mobilnih uređaja.

U CLDC-u je definisan skup interfejsa koji je namenjen osnovnoj mrežnoj nameni podataka, a specifične implementacije tih interfejsa su premeštene u MIDP API. Dakle, CLDC API se logički može podaliti na dva dela:

- Skup paketa koji je podskup J2SE API-ja
- Skup generičkih mrežnih interfejsa

Najveći deo klasa i interfejsa CLDC API-ja je direktno nasleđen iz J2SE API-ja. J2ME nameće ograničenje po kojem se ni jedna klasa ili interfejs koji su direktno preuzeti iz J2SE ne sme ni na koji način menjati, što pak znači da su metodi i polja identični verzijama koje postoje u J2SE. Na taj način je olakšano učenje programiranja pomoću J2ME, a takođe je postignuto da se programski kod lakše prenosi iz J2SE u J2ME i obrnuto.

CLDC se od J2SE API-ja razlikuje po svojim mogućnostima mrežne razmene podataka, što je istaknuto generičkim mrežnim okruženjem (Generic Connection Framework (GCF)). Namena GCF-a jeste definisanje opšte mrežne arhitekture koja omogućava mrežnu razmenu podataka (I/O) i neverovatno je fleksibilan, pa stoga je i proširiv. GCF je napravljen kao funkcionlan podskup J2SE klasa za mrežnu razmenu podataka, što znači da osobine koje su opisane u GCF-u postoje i u J2SE. GCF se uglavnom sastoji od skupa interfejsa za uspostavljanje veze, kao i klase `Connector` koja se koristi za uspostavljanje različitih veza. `Klasa Connector` i interfejsi za uspostavljanje veze se nalaze u paketu `java.microedition.io`. Mnogo više o programiranju mrežnih video-igara ćete naučiti u Poglavlju 14, „Osnove mrežne komunikacije između video-igara“.

MIDP API

Profil uređaja sadrži detaljne informacije o mogućnostima uređaja koje ne postoje u konfiguraciji kako bi mogle da se obave važni poslovi na zadatom tipu uređaja. U slučaju Mobile Information Device Profile (MIDP), tip uređaja je bežični mobilni uređaj, kakav je mobilni telefon ili pejdžer. Zbog toga je namena MIDP API-ja da na osnovu CLDC API-ja nadogradi neophodne klase i interfejse kako bi omogućio pravljenje MIDlet video-igara.

Slično CLDC API-ju, MIDP API se može podeliti na dva dela:

- Dve klase koje su direktno preuzete iz J2SE API-ja
- Skup paketa koji sadrže klase i interfejse koji su karakteristični za MIDP programiranje

MIDP API, kao i CLDC API, sadrži elemente koji su preuzeti iz J2SE API-ja. Većina novih MIDP API klasa i interfejsa su napravljeni za MIDP programiranje, što i nije iznenađenje. Mada ove klase i interfejsi mogu imati sličnu ulogu kao i neke klase i interfejsi koji postoje u J2SE API-ju, oni su jedinstveni za MIDP API i specijalno su napravljeni za rešavanje MIDlet problema. Ovaj deo MIDP API-ja je podeljen u nekoliko paketa. Imena ovih paketa u imenu imaju prefiks `javax.microedition:`

- `javax.microedition.midlet`
- `javax.microedition.lcdui`
- `javax.microedition.lcdui.game`
- `javax.microedition.media`
- `javax.microedition.media.control`
- `javax.microedition.io`
- `javax.microedition.pki`
- `javax.microedition.rms`

Paket `javax.microedition.midlet` jeste središnji paket MIDlet API-ja i sadrži samo jednu klasu: klasu `MIDlet. Klasa MIDlet` obezbeđuje osnovnu funkcionalnost koja je neophodna za MIDP aplikaciju (`MIDlet`) koja se može izvršavati na mobilnom uređaju. Više o klasi `MIDlet` ćete naučiti kako budete napredovali čitajući ovu knjigu i kako budete pravili složene MIDlet primere i video-igre.

Paketi `javax.microedition.lcdui` i `javax.microedition.lcdui.game` sadrže klase i interfejse za rad sa GUI komponentama koje su namenjene malim ekranima mobilnih uređaja. Pored toga, postoje klase i interfejsi koji su namenjeni pravljenju video-igara za mobilne uređaje. Jedinstvene osobine, kao što je animacija sprajtova i upravljanje slojevima, čine ove pakete veoma korisnim za programiranje video-igara za

mobilne uređaje. Kasnije u ovom poglavlju ćete se bliže upoznati sa klasama ovih paketa, a sticanje znanja ćete nastaviti kasnije u ovoj knjizi.

IGRAČKA RADIONICA

Ukoliko ste već koristili J2ME, onda će vas sigurno zanimati da je paket `javax.microedition.lcdui.game` *potpuno nov u MIDP 2.0. To je razlog zbog čega je MIDP 2.0 značajan skok kako bi J2ME postao značajna tehnologija za pravljenje video-igara za mobilne uređaje.*

Paketi `javax.microedition.media` i `javax.microedition.media.control` sadrže klase i interfejse za rad sa audio zapisima u MIDletu. Ovi paketi predstavljaju MIDP 2.0 Media API. MIDP 2.0 Media API je podskup većeg Mobile Media API-ja. Kompletan Mobile Media API podržava širok spektar objekata, kao što su slike, zvuk, muzika i video. Osobine MIDP 2.0 Media API-ja su ograničene na generisanje tonova i reprodukciju digitalnih zvučnih efekata iz WAV datoteka. Specifičnosti koje su vezane za reprodukovanje zvuka u MIDlet video-igrama ćete upoznati u Poglavlju 8, „Pravljenje buke pomoću tonova“.

Ranije ste naučili da CLDC predstavlja osnovu za mrežnu razmenu podataka i I/O pomoću Generic Connection Frameworka (GCF). MIDP API nadograđuje podršku koja je ugrađena u paket `javax.microedition.io`, koji pak sadrži nekoliko interfejsa i klasa za uspostavljanje bežičnih mrežnih veza i razmenu podataka. Paket `javax.microedition.pki` se koristi zajedno sa paketom javax.microedition.io kako bi se napravila bezbedna mrežna komunikacija. Osnovne poslove mrežne komunikacije ćete naučiti u Poglavlju 14, „Osnove mrežne komunikacije između video-igara“.

Pošto u mobilnim telefonima ne postoje hard diskovi niti sistem datoteka (za sada), verovatno nećete moći da u datoteke zapisujete stalne MIDlet podatke. Umesto toga MIDP API opisuje potpuno nov koncept zapisivanja i čitanja stalnih MIDlet podataka: Record Management System (RMS). MIDP RMS obezbeđuje jednostavnu API slogovnu bazu podataka u koju se zapisuju stalni podaci, recimo, spisak najboljih rezultata i podaci video-igre. Klase i interfejsi koji sačinjavaju RMS se nalaze u paketu `javax.microedition.rms`.

Razumevanje MIDleta

Kompanija Sun Microsystem koristi nastavak „let“ kako bi imenovala nekoliko tipova programa koji pripadaju Java tehnologiji. Appleti, servleti, spotleti i MIDleti su smo neki od primera. Kompanija Sun je htela da upotrebi termin „singlets“, ali je taj termin već upotrebio komičar Rič Hol (Rich Hall) u televizijskom zabavnom programu stanice HBO 1980-ih koji se zvao Not Necessarily the News. MIDleti su programi koji se prave pomoću J2ME API-ja i koji se koriste u mobilnom računarskom okruženju. Ovo okruženje se u osnovi sastoji od upravljača aplikacijama koji obezbeđuje način izbora i pokretanja MIDleta na mobilnom uređaju. Upravljač aplikacijama za MIDlet je odgovoran za dodeljivanje procesorskog vremena MIDletu.

Struktura MIDleta

Verovatno se nećete previše iznenaditi kada saznate da se svaki MIDlet mora izvesti iz standardne klase koja je deo MIDP API-ja. Klasa iz koje se MIDlet izvodi se nalazi u paketu `javax.microedition.midlet` i zove se `MIDlet`. Mada je u klasi MIDlet definisano samo nekoliko metoda, ti metodi su veoma važni za MIDlete koje ćete praviti. Metodi koji postoje u klasi MIDlet su:

- `startApp()` - pokreće MIDlet
- `pauseApp()`- zaustavlja MIDlet
- `destroyApp()`- uklanja MIDlet

Da biste bolje razumeli kakav uticaj ovi metodi imaju na MIDlet, morate zapamtiti da se MIDlet može nalaziti u tri različita stanja koja određuju kako on funkcioniše. Ta stanja su: aktivan, zaustavljen i uklonjen. Ova stanja direktno odgovaraju metodima klase MIDlet. Metode klase MIDlet direktno poziva upravljač aplikacijama okruženja u kojem se MIDlet izvršava. U nekim slučajevima ćete pozivati ove metode, a načešće ćete pozivati metod `destroyApp()`. Ovi metodi se jednim imenom nazivaju metodima životnog ciklusa pošto se pomoću njih upravlja životnim ciklusom MIDleta. Pomoću ovih metoda upravljač aplikacijama upravlja MIDletima i dodeljuje im pristip resursima uređaja.

Životni ciklus MIDleta

Životni ciklus MIDleta se odvija između tri stanja za koja ste upravo saznali. Uobičajeni MIDletu se najveći deo vremena nalazi u aktivnom stanju (Active) ili je zaustavljen (Pause), a zatim kada se MIDlet zatvori, prelazi u stanje uklanjanja (Destroy) sve dok se u potpunosti ne ukloni iz memorije. U većini MIDlet video-igara ćete zaobilaziti ova stanja, jer je veoma važno rezervisati i oslobađati resurse na osnovu stanja u kojem se MIDlet nalazi. Na primer, prilikom pokretanja MIDleta video-igre ćete verovatno morati da napravite objekte i/ili učitate podatke. Kada se izvršavanje MIDleta zaustavi, moraćete oslobodite neke resurse, prekinuti mrežne veze koje su uspostavljene i zaustaviti muziku video-igre. Najzad, posle uklanjanja MIDleta iz memorije, moraćete da oslobodite resurse koje je video-igra koristila, kao i da zapišete podatke o video-igri.

Zapamtite da MIDlet može nekoliko puta za svog postojanja ući u aktivno stanje ili biti zaustavljen. Međutim, pošto MIDlet uđe u stanje Destroyed, iz tog stanja se ne može vratiti. Zbog toga MIDlet ima samo jedan život.

Komande MIDleta

Pored zaobilaženja metoda životnog ciklusa, u većini MIDleta je implementiran metod `commandAction()`. Metod `commandAction()` je metod koji se pokreće događajem, a definisan je u interfejsu `javax.microedition.lcdui.CommandListener`. Komande se koriste za upravljanje MIDlet video-igrom i za iniciranje akcija kao što su zaustavljanje video-igre, zapisivanje podataka video-igre, podešavanja i prekidanje video-igre. MIDlet komande se mogu zadavati pomoću specijalnih tastera ili menija i moraju se obraditi pomoću metoda `commandAction()`.

IGRAČKA RADIONICA

Specijalni tasteri se nalaze pored ekrana mobilnog telefona i koriste se za zadavanje komandi koje su specifične za MIDlet. Ukoliko pritisnete specijalni taster zadaćete komandu koja se na ekranu prikazuje iznad tog tastera. Tasteri koji se koriste za upravljanje video-igre se obrađuju drugačije od komandi koje se njima zadaju, što ćete naučiti u Poglavlju 6, „Obrada akcija korisnika“.

Klase Display, Screen i Canvas

Još jedan važan MIDlet koncept koji treba u ovom trenutku pomenuti jeste klasa `Display`. Klasa `Display` se koristi za upravljanje ekranom uređaja. Klasa `Display` je definisana u paketu javax.microedition.lcdui (u istom paketu su definisane i druge `GUI` klase), a koristi se za upravljanje ekranom i komandama koje korisnik zadaje pomoću uređaja.

Nikada nećete praviti objekat klase `Display`. Obično ćete u metodu `startApp()` MIDlet video-igre koristiti pokazivač na objekat klase Display, a zatim ćete ga koristiti za uspostavljanje ekrana video-igre i korisničkog interfejsa. Postoji samo jedna instanca objekta klase `Display` za svaki MIDlet koji se izvršava u uređaju.

Druga veoma važna klasa koja je definisana u paketu `javax.microedition.lcdui` jeste klasa `Canvas`. Klasa Canvas predstavlja apstraktnu površinu veličine ekrana uređaja po kojoj se iscrtavaju elementi koje MIDlet prikazuje. Ova površina se koristi za direktno obavljanje grafičkih operacija kao što su crtanje linija i krivih ili prikazivanje slika. Površine za crtanje, kao što pretpostavljate, predstavljaju osnovu ekrana video-igara. Zapravo, postoji specijalna klasa `GameCanvas` koja je definisana u paketu `javax.microedition.lcdui`. Klasa `GameCanvas` se isključivo koristi za efikasno iscrtavanje grafike video-igara. Klasa `GameCanvas` se razlikuje od klase `Canvas`, jer sadrži veoma efikasne načine za iscrtavanje animirane grafike koja se veoma često koristi u video-igrama.

IGRAČKA RADIONICA

Ukoliko napravite video-igru koja se može podešavati ili se na neki drugi način od korisnika dobijaju informacije, onda ćete verovatno morati da koristite klasu `Screen` *koja je definisana u paketu* `javax.microedition.lcdui`. *Klasa* `Screen` *je generička* `MIDlet GUI` *komponenta koja nam služi kao osnovna klasa na osnovu koje se prave druge važne komponente. Ekrani su značajni jer daju kompletnu informaciju o ekranu. U bilo kom trenutku se može prikazati samo jedan ekran. Više ekrana možete zamisliti kao špil karata koje otvarate jednu po jednu. Većina MIDleta koristi podklase klase* `Screen`. *Podklase koje se koriste su* `Form`, `TextBox` *ili* `List` *(i sve su definisane u paketu* `javax.microedition.lcdui`*). Ove klase se koriste jer su specijalizovane. Ekrani se mogu koristiti zajedno sa klasom* `Canvas` *kako bi se napravio razumljiv* `GUI` *za* `MIDlet` *video-igre. Ekran i površinu za crtanje ne možete istovremeno prikazati, ali ih možete naizmenično prikazivati.*

Osnove pravljenja MIDleta

Pre nego počnete da pravite MIDlete morate instalirati J2ME Wireless Toolkit (koji ćete pronaći na propratnom CD-ROM-u). Takođe, možete koristiti i neki drugi Java alat za programiranje ukoliko pravite video-igru za određeni model mobilnih telefona, ali za sada ipak koristite J2ME Wireless Toolkit ukoliko želite da emulirate Java mobilne telefone na PC računaru.

Da biste MIDlet video-igru pretočili iz koncepta u stvarnost potrebno je da uradite sledeće:

1. Napravite datoteke sa izvornim programskim kodom.
2. Kompajlirajte datoteke sa izvornim programskim kodom u bajtkod klase.

3. Proverite bajtkod klase.
4. Bajtkod klase spakujte u JAR (Java ARchive) datoteku. U istu datoteku smestite dodatne resurse i manifest datoteku (uskoro ću vam objasniti šta ta datoteka treba da sadrži).
5. Napravite JAD (deskriptor aplikacije) datoteku koja će pratiti JAR datoteku.
6. Testirajte i debagirajte MIDlet.

Prvi korak možete obaviti pomoću jednostavnog tekstualnog editora. Ukoliko nemate programski tesktualni editor možete upotrebiti jednostavan editor Notepad koji postoji u Windowsu. U koraku 2 morate upotrebiti standardni Java kompajler kako biste datoteke sa izvornim programskim kodom kompajlirali za MIDlet. U koraku 3 treba da proverite kompajlirane bajtkod klase pomoću specijalnog alata. U koraku 4 pomoću alatke Java archive (JAR) treba da kompresujete MIDlet datoteke sa izvornim programskim kodom i da u JAR datoteku dodate manifest datoteku. U koraku 5 treba da napravite specijalnu datoteku sa opisom aplikacije. To je tekstualna datoteka koja sadrži infromacije o MIDletu. I, najzad, u koraku 6 možete da uživate i testirate MIDlet pomoću J2ME emulatora.

Mada svaki od navedenih koraka možete obaviti pomoću alatki koji se koriste sa komande linije (deo su paketa J2ME Wireless Toolkit), u prethodnom poglavlju ste saznali koliko alat KToolbar kompanije Sun pojednostavljuje postupak pravljenja/testiranja MIDleta.

Pravljenje programa Skeleton

Mada bih ja najviše voleo da vam odmah pokažem kako se pravi 3D video-igra za više igrača, ne verujem da bi mogla da mi posluži kao najbolji primer pomoću kojeg bih vam objasnio strukturu MIDleta i kako se programiraju zbog složenosti takve video-igre. Umesto toga, u ovom odeljku ćete naučiti kako da napravite veoma jednostavan MIDlet, MIDlet Skeleton, koji pomoću teksta prikazuje informacije o mobilnom telefonu. Pošto te informacije obuhvataju informacije o veoma važnim parametrima uređaja, kao što su veličina ekrana za video-igre i broj boja koji se može prikazati, koristićete ga za proveru osobina mobilnih telefona.

Prilikom pravljenja MIDleta Skeleton obavićete sve korake koje sam naveo u prethodnom odeljku. Ovaj postupak ćete, sa više ili manje izmena, koristiti za pravljenje svake MIDlet video-igre. Evo šta sve treba da uradite kako biste napravili video-igru Skeleton:

1. Treba da napišete programski kod MIDleta.
2. Treba da kompajlirate MIDlet.
3. Treba da proverite MIDlet.

4. Treba da upakujete MIDlet.
5. Treba da testirate MIDlet.

U nekoliko narednih odeljaka ćete saznati kako da obavite ove korake. Pošto ih sve obavite napravićete svoj prvi J2ME MIDlet.

Pisanje programskog koda

U ovom odeljku ćete, deo po deo, napisati programski kod MIDleta Skeleton. Kompletan programski kod ćete videti na kraju poglavlja. Prvi deo koda MIDleta služi za uvoz nekoliko važnih J2ME paketa. Mada možete izbeći uvoz bilo kog paketa i svaki interfejs i klasu J2ME referencirati navođenjem potpuno kvalifikovanog imena (`javax.microedition.midlet.MIDlet`, na primer), takvo navođenje može postati zamorno i na kraju dovodi do programskog koda koji je teško razumljiv. Dakle, prva dva programska reda vašeg MIDleta uvoze dva osnovna paketa koja se koriste pri MIDlet programiranju:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
```

PODSETNIK

*Većina Java programera uvoze ceo paket koristeći džoker znak *, jer se na taj način ne otkriva mnogo o klasi koju uvozite. Međutim, to je brz i lak način da se uvezu sve klase koje postoje u paketu, a što se knjige tiče, ja sam se potrudio da sve bude što je jednostavnije moguće. Klase možete uvoziti jednu po jednu kako bi programski kod bio razumljiviji i eksplicitniji.*

Paket javax.microedition.midlet obezbeđuje podršku za klasu MIDlet, dok paket javax.microedition.lcdui sadrži podršku za GUI klase i interfejse koji se koriste za pravljenje MIDlet GUI-a (sadrži, recimo, klasu Display). Pošto uvezete ova dva paketa možete deklarisati klasu SkeletonMIDlet koju izvodite iz klase MIDlet:

```
public class SkeletonMIDlet extends MIDlet implements CommandListener {
```

Činjenica da klasa `SkeletonMIDlet` predstavlja proširenje klase MIDlet nije nikakvo iznenađenje, ali je implementacija interfejsa `CommandListener` pomalo neobična. Ovaj interfejs je neophodan za pravljenje komande Exit pomoću koje korisnik izlazi iz MIDleta. Tačnije, interfejs `CommandListener` je implementiran tako da MIDlet može da odgovori na događaje komandi.

Jedina promenljiva članica klase `SkeletonMIDlet` je objekat `SCanvas` koji predstavlja glavni ekran:

```
private SCanvas canvas;
```

Klasa `SCanvas` je prilagođena `MIDlet` klasa površine za crtanje koja je, što ćete uskoro saznati, izvedena iz klase `Canvas`. Površina za crtanje se inicijalizuje pomoću metoda `startApp()`:

Komanda Exit je napravljena i pridodata površini za crtanje tako da na nju može da odgovori.

```
public void startApp() {
   if (canvas == null) {
      canvas = new SCanvas(Display.getDisplay(this));
      Command exitCommand = new Command("Exit", Command.EXIT, 0);
      canvas.addCommand(exitCommand);
      canvas.setCommandListener(this);
   }
   // Start up the canvas
   canvas.start();
}
```

Metod `startApp()` se poziva svaki put kada MIDlet pređe u stanje Active, a prvo što metod startApp() uradi jeste stvaranje površine za crtanje. Objekat klase `Display` MIDleta dobija i prosleđuje površini za crtanje kao deo postupka pravljenja. Zatim se pravi komanda `Exit`. Komanda se pravi kada konstruktoru Command prosledite tri parametra kojima zadajete ime komande, njen tip i prioritet. Ime komande zadaje korisnik i ono se pojavljuje iznad jednog od specijanih tastera mobilnog telefona ili u meniju, što zavisi od prioriteta komande broja tastera koji se mogu upotrebiti. Tip komande može biti jedan od nekoliko tipova koji su predstavljeni ugrađenim konstantama konstruktora `Command` (recimo, `EXIT`, `OK` ili `CANCEL`).

Komanda se dodaje površini za crtanje kako bi postala aktivna. Morate odabrati osluškivača komandi koji će prihvatati i obrađivati komandne događaje. To ćete uraditi tako što ćete pozvati metod `setCommandListener()` i proslediti mu this. Na taj način ćete MIDlet klasu (zapravo, klasu `SkeletonMIDlet`) odrediti za osluškivača komandi. To je baš ono što nam treba, jer smo ranije u klasu implementirali interface `CommandListener`.

PODSETNIK

Prioritet komandi se koristi za određivanje redosleda komandi za korisnika. Zadavanje prioriteta je neophodno jer većina uređaja ima mali broj tastera koji se mogu koristiti u MIDletu. Zbog toga su samo najvažnije komande dodeljene trima specijalnim tasterima. Druge komade se svakako mogu koristiti, ali se mogu koristiti pomoću menija MIDleta do kojih nije lako doći kao do specijalnih tastera. Vrednost kojom se označava prioritet komande se smanjuje sa porastom prioriteta. U primeru Skeleton, komandi Exit je dodeljen prioritet 2, što pak znači da je njen prioritet veoma visok. Naravno, vrednosti prioriteta su potpuno relativne, a pošto u primeru nema drugih komandi onda ova vrednost nema neko naročito značenje.

Komanda Exit MIDleta Skeleton se obrađuje u metodu `commandAction()`:

```
public void commandAction(Command c, Displayable s) {
   if (c.getCommandType() == Command.EXIT) {
      destroyApp(true);
      notifyDestroyed();
   }
}
```

Dva argumenta koja se prosleđuju u metod commandAction() su komanda i ekran na kojem se komanda prikazuje. U ovom primeru nas interesuje samo komanda. Objekat `Command` se poredi sa konstantom članicom `Command.EXIT` kako bi se proverilo da je komanda `Exit` zaista komanda koja se obrađuje. Ukoliko jeste, poziva se metod `destroyApp()` kako bi se uklonio MIDlet. Argument true koji se prosleđuje metodu označava bezuslovno uklanjanje, što znači da je MIDlet uklonjen čak i kada se tokom uklanjanja dogodi greška ili izuzetak. Zatim se poziva metod `notifyDestroyed()` kako bi se upravljač aplikacijama obavestio da je MIDlet prešao u stranje Destroyed.

MIDlet Skeleton ne koristi metode `pauseApp()` i `destroyApp()`, ali ipak morate napraviti njihove prazne implementacije:

```
public void pauseApp() {}
public void destroyApp(boolean unconditional) {}
```

Iako ste delove programa videli pojedinačno, kompletan programski kod za SkeletonMIDlet.java se nalazi u listingu 3.1.

Listing 3.1 Programski kod klase SkeletonMIDlet koja se nalazi u datoteci SkeletonMIDlet.java.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class SkeletonMIDlet extends MIDlet implements CommandListener {
   private SCanvas canvas;

   public void startApp() {
      if (canvas == null) {
        canvas = new SCanvas(Display.getDisplay(this));
        Command exitCommand = new Command("Exit", Command.EXIT, 0);
        canvas.addCommand(exitCommand);
        canvas.setCommandListener(this);
   }
   // Start up the canvas
   canvas.start();
}
   public void pauseApp() {}

   public void destroyApp(boolean
   unconditional) {}

   public void commandAction(Command c, Displayable s) {
      if (c.getCommandType() == Command.EXIT) {
```

Ovi metodi se u primeru uopšte ne koriste, ali morate napraviti prazne implementacije kako bi se zadovoljila pravila klase MIDlet

```
        destroyApp(true);
        notifyDestroyed();
      }
    }
}
```

Potrebno je da pozovete metod destroyApp(), iako se zapravo pomoću metoda notifyDestroyed() uklanja MIDlet.

Preostali deo programskog koda primera Skeleton MIDlet se odnosi na klasu SCanvas, koja je prikazana u listingu 3.2.

Listing 3.2 Klasa SCanvas služi kao prilagođena površina za crtanje u primeru Skeleton MIDlet.

```
import javax.microedition.lcdui.*;

public class SCanvas extends Canvas {
   private Display display;

   public SCanvas(Display d) {
      super();
      display = d;
   }

   void start() {
      display.setCurrent(this);
      repaint();
   }

   public void paint(Graphics g) {
      // Clear the canvas
      g.setColor(0, 0, 0); // black
      g.fillRect(0, 0, getWidth(), getHeight());
      g.setColor(255, 255, 255); // white

      // Draw the available screen size
      int y = 0;
      String screenSize = "Screen size: " +
      Integer.toString(getWidth())
➥ + " x " +
        Integer.toString(getHeight());
      g.drawString(screenSize, 0, y, Graphics.TOP ¦ Graphics.LEFT);

      // Draw the number of available colors
      y += Font.getDefaultFont().getHeight();
      String numColors = "# of colors: " +
➥Integer.toString(display.numColors());

      g.drawString(numColors, 0, y, Graphics.TOP ¦ Graphics.LEFT);

      // Draw the number of available alpha levels
      y += Font.getDefaultFont().getHeight();
      String numAlphas = "# of alphas: " +
      Integer.toString(display.numAlphaLevels());
```

Ovaj red programskog koda je veoma važan jer ovu površinu za crtanje proglašava površinom po kojoj crta MIDlet.

Veoma je važno očistiti pozadinu pre nego što se iscrta sadržaj.

LISTING 3.2 Nastavak

```
      g.drawString(numAlphas, 0, y, Graphics.TOP ¦ Graphics.LEFT);
      // Draw the amount of total and free memory
      Runtime runtime = Runtime.getRuntime();
      y += Font.getDefaultFont().getHeight();
      String totalMem = "Total memory: " +
        Long.toString(runtime.totalMemory() / 1024) + "KB";
      g.drawString(totalMem, 0, y, Graphics.TOP ¦ Graphics.LEFT);
      y += Font.getDefaultFont().getHeight();
      String freeMem = "Free memory: " +
      Long.toString(runtime.freeMemory()
➥/ 1024) + "KB";

      g.drawString(freeMem, 0, y, Graphics.TOP ¦ Graphics.LEFT);
   }
}
```

Klasa `SCanvas` je izvedena iz klase `Canvas`, a jedini parametar u njenom konstruktoru je objekat `Display`. Zadatak konstruktora je da zada vrednost promenljivoj članici display tako da se ekran `MIDleta` može koristiti pomoću programskog koda površine za crtanje. `Metod start()` poziva metod `setCurrent()` za objekat `Display` koji površinu za crtanje postavlja za tekući ekran. U MIDletu može postojati više ekrana. U tom slučaju koristite metod `setCurrent()` kako biste prelazili sa ekrana na ekran. Metod `start()` poziva metod repaint() kako bi se na površini pravilno iscrtala slika.

IGRAČKA RADIONICA

Mada je klasa SCanvas MIDleta Skeleton izvedena iz klase Canvas, većina primera u knjizi koristi klase površina za crtanje koje su izvedene iz klase GameCanvas koja ima osobine koje su specifične za video-igre (recimo, dvostruko baferisanu grafiku i efikasnu obradu događaja tastature). Te osobine ne koristimo u primeru Skeleton.

Veći deo programskog koda primera Skeleton se odnosi na iscrtavanje po površini za crtanje i obavlja se pomoću metoda `paint()`. Za sada ne morate razumeti sve detalje koji su vezani za programski kod za crtanje - u narednom poglavlju ću detaljno objasniti grafiku video-igara za mobilne uređaje. Međutim, ukratko ću vam objasniti kako se crta ukoliko želite to unapred da saznate.

Pomoću metoda se prvo uklanja sadržaj površine za crtanje, tako što se ona ispunjava bojom (u ovom slučaju se ispunjava crnom bojom). Crna boja se zatim menja u belu kako bi mogao da se ispiše tekst. Prvo se dobija veličina ekrana, a zatim se prikazuje tekst tako da je centriran pri vrhu ekrana. Zatim se saznaje broj raaspoloživih boja i alfa kanala i prikazuje se ispod veličine ekrana. Na kraju se prikazuje veličina ukupne i slobodne memorije.

Pošto smo napisali programski kod, gotovo da smo spremni da napravimo i testiramo MIDlet Skeleton. Međutim, potrebno je da prvo napravite nekoliko datoteka koje su neophodne za pravljenje paketa MIDleta.

IGRAČKA RADIONICA

Broj alfa kanala koju mobilni telefon podržava se određuje na osnovu toga koliko kontrole imate nad transparentnim oblastima slika. Na primer, svi mobilni telefoni imaju najmanje dva alfa nivoa koji odgovaraju pikselu koji je potpuno providan i pikselu koji je potpuno neprovidan. Neki mobilni telefoni imaju po 256 alfa nivoa, što vam omogućava da naprvite grafiku koja je delimično providna, dakle, kroz sliku se može videti različit intenzitet pozadine.

Priprema MIDleta za distribuiranje

Pakovanje MIDlet video-igre radi distribuiranja obuhvata kompresovanje raznih datoteka koje MIDlet koristi u arhivi koja se naziva JAR datoteka. Pored kompresovanja klase koja je proverena u JAR datoteku, JAR datoteka mora sadržati datoteke sa resursima koji se koriste u MIDletu, kao i manifest datoteka koja opisuje sadržaj JAR datoteke.

U slučaju MIDleta Skeleton jedini resurs je ikona koja se prikazuje pored imena MIDleta. Ikonu ću vam ubrzo objasniti. Za sada ćemo pažnju zadržati na manifest datoteci. *Manifest* datoteka je specijalna tekstualna datoteka koja sadrži spisak svojstava MIDleta i njihove vrednosti. Ove informacija su veoma bitne jer označavaju ime, ikonu i ime klase svakog MIDleta u JAR datoteci, kao i specifične CLDC i MIDP verzije koje koriste MIDleti koji se nalaze u JAR datoteci. Zapamtite da se više MIDleta može nalaziti u jednoj JAR datoteci. U tom slučaju se datoteka naziva MIDlet paket.

PODSETNIK

Svi primeri u knjizi koriste MIDP 2.0 i CLDC 1.0. Mada neki mobilni telefoni podržavaju CLDC 2.0, CLDC 1.0 je dovoljan za većinu video-igara za mobilne uređaje. Međutim, MIDP 2.0 je veoma važan jer su u ovu verziju ugrađene mnoge nove osobine koje se koriste pri pravljenju video-igara za mobilne uređaje.

Manifest datoteka MIDlet paketa mora imati ime `Manifest.mf` i mora se nalaziti u JAR datoteci zajedno sa MIDlet klasama i resursima. Evo programskog koda manifest datoteke MIDleta Skeleton:

```
MIDlet-1: Skeleton, /icons/Skeleton_icon.png, SkeletonMIDlet
MIDlet-Name: Skeleton
MIDlet-Description: Skeleton Example MIDlet
MIDlet-Vendor: Stalefish Labs
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-2.0
```

U prvom redu programskog koda manifest datoteke je najvažniji, jer se u njemu zadaje ime MIDleta, datoteka u kojoj se nalazi ikona i ime izvršne klase. Zapazili ste da je ime svojstva `MIDlet - 1`. Ukoliko će se u datoteci nalaziti više `MIDleta` onda ćete ih pozivati sa MIDlet - 2, MIDlet - 3 i tako dalje. Svojstva `MIDlet - Name`, `MIDlet - Description`, `MIDlet - Vendor` i `MIDlet - Version` se odnose na ceo MIDlet paket.

Međutim, u našem primeru MIDlet Skeleton je jedini MIDlet paketa pa se može ceo paket nazvati Skeleton. Poslednja dva svojstva označavaju verzije konfiguracija i profila za koje je MIDlet namenjen. U našem primeru su to CLDC 1.0 i MIDP 2.0.

Ranije sam napomenuo da u JAR datoteku morate smestiti sve resurse, MIDlet klase i manifest datoteku. MIDlet će kao resurs imati najmanje jednu ikonu. Ikona je slika veličine 12x12 piksela koja je zapisana u datoteci PNG formata. Ikona može biti u boji ili crno-bela, što zavisi od ekrana mobilnog telefeona za koji je MIDlet namenjen. Ja sam napravio malu ikonu lobanje koja će poslužiti kao ikona primera Skeleton, a zapisana je u datoteci `Skeleton_icon.png`.

Jedno upozorenje koje se tiče ikone za MIDlet: datoteka sa ikonom mora biti smeštena u direktorijum icons unutar JAR datoteke MIDleta. JAR datoteke, slično ZIP datotekama, omogućavaju postojanje direktorijuma unutar arhive. Da biste datoteku smestili u jedan od direktorijuma koji postoji unutar arhive, morate je referencirati iz poddirektorijuma prilikom dodavanja u JAR datoteku. MIDlet resursi se standardno zapisuju u direktorijum res, ispod direktorijuma u koji su smeštene datoteke sa programskim kodom. Znajući to, jednostavno rešenje za zapisivanje ikone jeste da direktorijum icons smestite unutar direktorijuma res.

Slika 3.2
Držite se strukture direktorijuma za dato-teke sa programskim kodom MIDleta.

Kada već pominjem strukturu direktorijuma i MIDlete, postoji standardni način za organizovanje direktorijuma za datoteke sa izvornim kodom MIDleta. Na slici 3.2 vidite strukturu direktorijuma koje treba da se držite kada raspoređujete datoteke vašeg MIDleta.

Različiti direktorijumi koje vidite na slici se koriste za:

- `src` - datoteke sa Java programskim kodom
- `bin` - manifest datoteka, JAD datoteka i JAR datoteka
- `classess` — datoteke koje sadrže kompajliran Java bajtkod
- `tmpclassess` - datoteke koje sadrže kompajliran i proveren Java bajtkod
- `res` - datoteke u kojima se nalaze resursi, ne računajući datoteke sa ikonama (slike, zvuci i tako dalje)
- `res/icons` - datoteke koje sadrže ikone

PODSETNIK

Primeri MIDleta koje dobijate uz J2ME Wireless Toolkit, uključujući primere igara koje ste videli u prethodnom poglavlju, drže se ove strukture direktorijuma.

Pored manifest datoteke koja se nalazi u JAR datoteci MIDleta, neophodno je da postoji datoteka sa deskriptorom aplikacije. Datoteka deskriptora aplikacije (engl. *applica-*

tion descriptor), odnosno JAD datoteka, sadrži informacije koje su slične informacijama koje se nalaze u manifest datoteci. JAD datoteka koristi Java emulator prilikom testiranja paketa. Evo kako izgleda JAD datoteka MIDleta Skeleton:

```
MIDlet-1: Skeleton, /icons/Skeleton_icon.png, SkeletonMIDlet
MIDlet-Name: Skeleton
MIDlet-Description: Skeleton Example MIDlet
MIDlet-Vendor: Stalefish Labs
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-2.0
MIDlet-Jar-Size: 2706
MIDlet-Jar-URL: Skeleton.jar
```

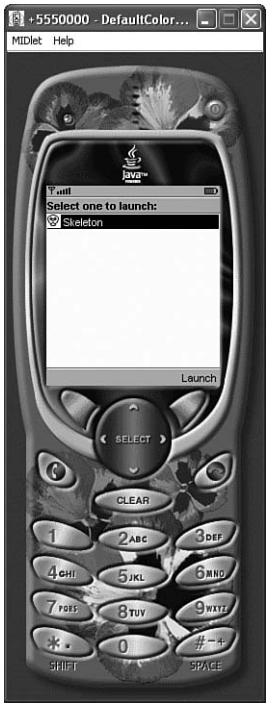
Ovo je broj verzije MIDleta. U našem primeru je to MIDlet 1.0.

Ukoliko ova veličina (u bajtovima) ne odgovara veličini JAR datoteke, MIDlet nećete moći pokrenuti

Sa izuzetkom poslednja dva elementa, JAD datoteka sadrži informacije sa kojima ste već upoznati. Poslednja dva elementa se odnose na veličinu JAR datoteke (u bajtovima) i ime JAR datoteke. Veoma je važno da ažurirate veličinu JAR datoteke svaki put kada prepakujete MIDlet jer će se veličina verovatno promeniti prilikom svakog pakovanja MIDleta.

Pravljenje i testiranje gotovog proizvoda

U prethodnom poglavlju sam vam pomenuo alat KToolbar. Alat KToolbar vam omogućava da napravite i pokrenete MIDlete uz veoma malo truda. Da biste pomoću alata KToolbar napravili MIDlet Skeleton prvo morate kopirati ceo direktorijum MIDleta Skeleton u direktorijum aplikacije u koji je instaliran paket J2ME Wireless Toolkit. Pošto kopirate direktorijum aplikacije Skeleton, projekat Skeleton možete otvoriti u alatu KToolbar tako što ćete kliknuti dugme Open Project koje se nalazi na paleti alatki.

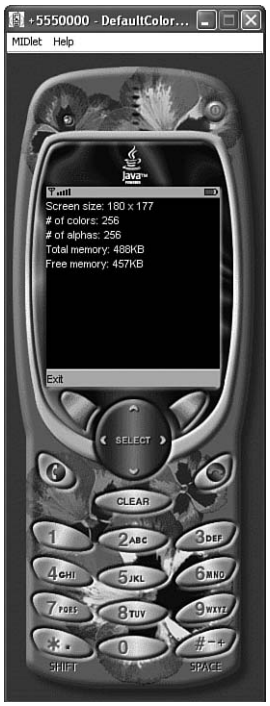


Slika 3.3
MIDlet Skeleton se može pokrenuti u upravljaču aplikacijama emulatora J2ME.

Pošto otvorite projekat potrebno je da kliknete dugme Build, koje se nalazi na paleti alatki, kako biste napravili MIDlet Skeleton. Pošto napravite projekat, kliknite dugme Run (koje se takođe nalazi na paleti alatki) kako biste pokrenuli aplikaciju u emulatoru J2ME. Na slici 3.3 vidite da se MIDlet Skeleton može pokrenuti u emulatoru.

PODSETNIK

Programski kod i odgovarajuće datoteke primera MIDleta koje ćete videti u knjizi se nalaze na propratnom CD-ROM-u.



Slika 3.4
MIDlet Skeleton prikazuje informacije o mobilnom telefonu, kao što su veličina ekrana i broj boja.

Pošto je MIDlet Skeleton jedini MIDlet u paketu, on je već označen i može se odmah pokrenuti. Da biste ga pokrenuli potrebno je da kliknete dugme Action koje se nalazi na tastaturi uređaja, ili dugme specijalan taster Launch. Takođe, možete pritisnuti taster Enter na tastaturi računara. Na slici 3.4 vidite kako se MIDlet Skeleton izvršava u emulatoru.

Da biste izašli iz MIDleta Skeleton kliknite specijalan taster koji se nalazi ispod reči Exit. Na taj način ćete zadati komandu Exit čime ćete prekinuti izvršavanje MIDleta. Izvršavanje MIDleta možete prekinuti i kada kliknete crveno dugme End, koje se na mobilnom telefonu koristi za prekidanje veze.

Rezime

U ovom poglavlju ste napokon videli Java programski kod MIDleta. Mada MIDlet nije bio video-igra, morali ste da ga napravite počinjući od nule i datoteke sa programskim kodom kako biste napravili JAR datoteku. Pre nego što sam se u ovom poglavlju posvetio pisanju programskog koda, ja sam vam objasnio koje delove ima MIDlet, a dao sam vam i opšte osnove J2ME API-ja. Saznali ste kakav je životni ciklus MIDleta i koji metodi klase MIDlet se koriste za upravljanje životnim ciklusom.

U ovom poglavlju ste se upoznali sa nekoliko klasa i interfejsa koji su jedinstveni za Java programiranje za mobilne telefone. U narednom poglavlju ćemo se približiti programiranju video-igara za mobilne telefone tako što ćete se upoznati sa grafikom.

Renoviranje video-igre

Želim da krenete malim koracima kako biste se navikli na pravljenje MIDleta. Dakle, prva prepravka je veoma jednostavna. Treba da obavite sledeće korake kako biste promenili ikonu MIDleta Skeleton:

1. Napravite novu ikonu za primer MIDleta Skeleton. Obavezno je zapišite u PNG formatu, mora biti veličine 12x12 piksela i mora se nalaziti u direktorijumu res/iscons.
2. Izmenite manifest datoteku i JAD datoteku tako da MIDlet koristi novu ikonu.
3. Ponovo izgradite MIDlet i testirajte ga pomoću alata KToolbar.

Ukoliko sve bude u redu videćete da se ikona MIDleta promenila kada ga budete prikazali u J2ME emulatoru.

blanko