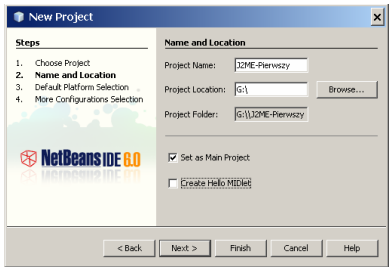
INSTRUKCJA DO ĆWICZENIA 4

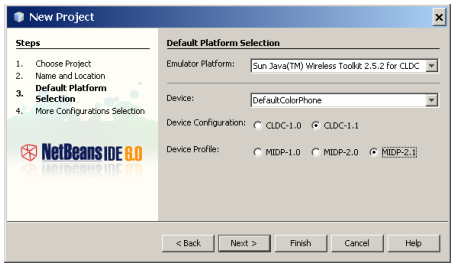
PIERWSZA APLIKACJA W JAVIE NA TELEFON KOMÓRKOWY

INTERFEJS WYSOKIEGO POZIOMU KLASY SCREEN

I Tworzenie pierwszego MIDletu przy pomocy środowiska NetBeans.

1. Otwórz nowy projekt w środowisku **NetBeans**, wybierając z menu **File** opcję **New Project…** a następnie zaznacz w panelu *Categories*: **Java ME,** w panelu *Project*: **Mobile Application.** Naciśnij przycisk **Next >.**
2. Projekt nazwij **J2ME-Pierwszy.** Określ położenie projektu *Project Location*: przyciskiem **Browse…** wybierając folder **G:**
3. Koniecznie **usuń zaznaczenie** opcji *Create Hello MIDlet* (patrz poniżej), a pozostaw zaznaczenie opcji *Set as Main Project*.
4. Naciśnij przycisk **Next >.**





5. W trzecim kroku tworzenia nowego projektu w oknie **New Project,** wybierz w okienku dialogowym *Emulator Platform*: Sun Java™ Wirless Toolkit 2.5.2 for CLDC, następnie sprawdź, czy ustawienia: typ urządzenia, konfiguracji i profili są zgodne z tym pokazanym na rysunku wyżej (DefaultColorPhone, CLDC-1.1, MIDP-2.1) jeśli tak, naciśnij przycisk **Finish.**
6. W okienku *Projects* kliknij nazwę projektu **J2ME-Pierwszy** prawym przyciskiem myszy a następnie wybierz opcję **New** i dalej w otwartym oknie dialogowym wybierz **MIDlet….**
7. Zmień nazwę *MIDlet Name*: na **MIDletKlasyScreen**. Naciśnij przycisk **Finish.** W oknie edytora kodu automatycznie zostanie otwarty do edycji plik *MIDletKlasyScreen.java.*

II Podstawowy szkielet MIDetu – opis podstawowych wymagań

1. Nasz wygenerowany program powinien mieć następującą postać (pominięto komentarze):

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

public class MIDletKlasyScreen extends MIDlet {
    public void startApp() {
    }

    public void pauseApp() {
    }

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

2. Za deklaracją klasy głównej `MIDletKlasyScreen` a przed metodą `startApp()` wygeneruj automatycznie konstruktora klasy:

```
public MIDletKlasyScreen() {
}
```

Deklaracja koniecznych metod

Dziedziczenie po klasie MIDlet zobowiązuje nas do zdefiniowania trzech metod, które w klasie MIDlet zostały oznaczone jako abstrakcyjne. Metody abstrakcyjne nie zawierają żadnej treści i zmuszają programistów tworzących klasy pochodne do samodzielnego ich zdefiniowania.
Metody abstrakcyjne, o których mowa, to:

```
public void startApp()
public void pauseApp()
public void destroyApp(boolean unconditional)
```

Są one bezpośrednio związane ze stanem, w jakim znajduje się MIDlet.
W momencie uruchamiania aplikacji wywołana zostaje metoda `startApp()`. Jeśli chcemy zwolnić zaalokowane wcześniej zasoby, możemy użyć metody `destroyApp()` przed zakończeniem działania aplikacji.
Metoda `pauseApp()` zostaje ona wywołana w chwili, gdy ktoś dzwoni i telefon musi zająć się obsługą rozmowy.

III Projektowanie wyglądu interfejsu wysokiego poziomu

1. Przed konstruktorem wpisz instrukcję, która zadeklaruje istnienie ekranu klasy `Form` o nazwie `formatka:`

```
Form formatka = new Form("Pierwszy MIDlet");
```

2. Obiekt klasy **Display** jest logiczną reprezentacją ekranu telefonu komórkowego. Każdy MIDlet ma dedykowany dokładnie jeden taki obiekt i należy pozyskać referencję do niego poprzez wywołanie metody: `getDisplay()`. Aby to wykonać utwórzmy obiekt typu `Display` o nazwie `display` (przed konstruktorem) następująco:

```
Display display;
```

Teraz musimy przechować referencję do tego obiektu poprzez wywołanie metody `getDisplay()`. W tym przypadku najlepiej wpisać taką instrukcję w konstruktorze klasy, ponieważ ta metoda zostanie wykonana automatycznie jako pierwsza i tylko jeden raz.

```
display = Display.getDisplay(this);
```

3. Wyświetlimy na naszym ekranie tekst `Witamy społeczność akademicką!` Najłatwiej tego dokonać wpisując na końcu w konstruktorze głównej klasy instrukcję, która doda do ekranu o nazwie `formatka` tekst:

```
formatka.append("Witamy społeczność akademicką!");
```

4. Uruchom aplikację naciskając klawisz <F6>. Naciśnij przycisk opcji **Launch**. Dlaczego brak jest jakiejkolwiek reakcji na wybranie opcji **Launch**. Zamknij emulator przyciskiem ✖ znajdującym się w prawym górnym narożniku okna.

5. Przyczyną takiego stanu rzeczy jest brak ustalenia, jaki ekran ma być wyświetlany na `display`. W tym przypadku należy wpisać instrukcję w metodzie `startApp()`, która określi że, aktualnie wyświetlanym ekranem ma być ekran o nazwie `formatka`.

```
display.setCurrent(formatka);
```

6. Uruchom aplikację.

IV Zamknięcie aplikacji poleceniem Command

W programach pisanych na telefony komórkowe nie uświadczymy tradycyjnych przycisków (*Button*), ich funkcje doskonale spełniają polecenia (*Command*). Na ekranie znajdują się one nad przyciskami opcji i to właśnie te przyciski wywołują polecenia. Po utworzeniu odpowiedniego polecenia trzeba je jeszcze dodać do formatki.

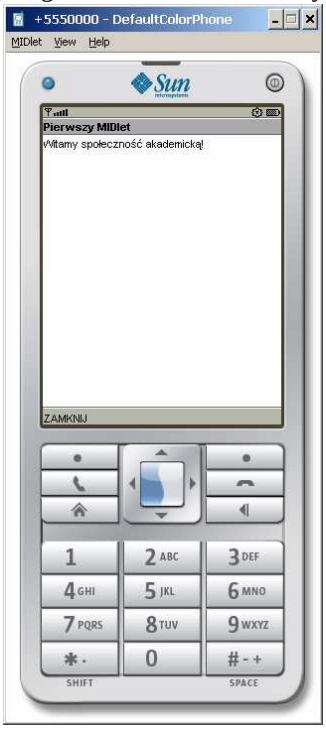
Zmodyfikuj tak program, aby po wypisaniu tekstu na ekranie zamknięcie aplikacji nastąpiło po wybraniu opcji ZAMKNIJ

1. Jeśli więc chcemy operować komendami (poleceniami) w naszej aplikacji, jedna z naszych klas musi implementować interfejs `CommandListener`. Zatem dodaj do klasy `MIDletKlasyScreen` interfejs **CommandListener**, który pozwoli nam operować komendami obsługujące przyciski opcji. W tym celu zmień nagłówek klasy `MIDletKlasyScreen` do postaci

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

2. Określenie obiektu słuchacza zdarzeń, który implementuje interfejs `CommandListener` wymaga zdefiniowania metody `commandAction();` którą w naszym przypadku umieścimy za metodą `destroyApp()`

```
public void commandAction(Command c, Displayable d) {
}
```



3. Dodaj poniższą instrukcję przed konstruktorem klasy `MIDletKlasyScreen`

```
Command wyjscie = new Command("ZAMKNIJ", Command.EXIT,1);
```

4. Wewnątrz konstruktora klasy `MIDletKlasyScreen` dopisz instrukcje dodające komendę oraz włącz nasłuchiwanie zdarzeń z ekranu:

```
// dodaj komendę
formatka.addCommand(wyjscie);
formatka.setCommandListener(this);
```

5. W metodzie `commandAction ()` możemy rozpoznać wywołaną komendę i aktywny ekran `Displayable` i odpowiednio na nią zareagować.

```
public void commandAction(Command c, Displayable d) {
    if (c == wyjscie) {
        destroyApp(true);
        notifyDestroyed();
    }
}
```

6. Uruchom aplikację <F6>. Sprawdź działanie polecenia ZAMKNIJ.

DRUGA APLIKACJA W JAVIE NA TELEFON KOMÓRKOWY
INTERFEJS NISKIEGO POZIOMU KLASY CANVAS

V Tworzenie drugiego MIDletu przy pomocy środowiska NetBeans.

1. Otwórz nowy projekt typu **Mobile Application** o nazwie **J2ME-Drugi** na dysku **G:** Nadaj nazwie MIDletu nazwę **MIDletKlasyCanvas**. Czynności wykonaj zgodnie z instrukcją (pkt.: *I*.1-7).

VI Podstawowy szkielet MIDetu – projektowanie wyglądu interfejsu niskiego poziomu

1. Zbudujmy **wewnętrzną klasę** `EkranG` rozszerzającą klasę `Canvas`, zaraz na początku części publicznej klasy `MIDletKlasyCanvas`. Zatem umieść poniższy kod wyróżniony szarym tłem (*patrz następna strona*):

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

public class MIDletKlasyCanvas extends MIDlet {
```

```
    private class EkranG extends Canvas {

        public EkranG(String txtlab){
        }
        protected void paint(Graphics g){
        }
    }

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

7. Dokonaj przeformatowania kodu programu <Alt+Shift+F>. (lub kliknij prawym przyciskiem wewnątrz edytora kodu i wybierz opcję **Format**.)
8. Przed konstruktorem klasy `EkranG` umieść deklaracje trzech zmiennych:

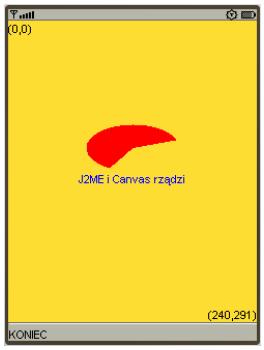
```
String etykieta = "Tekst domyślny";//tekst do wypisania na ekranie
int szerokośćElipsy = 90;           //szerokość elipsy na środku ekranu
int szerokośćEkranu, wysokośćEkranu;  //szerokość i wysokość ekranu
```

9. Wewnątrz konstruktora klasy `EkranG` umieść poniższe instrukcje:

```
etykieta = txtlab;
//Pobieram wysokość i szerokość dostępnego ekranu:
szerokośćEkranu = getWidth();
wysokośćEkranu = getHeight();
```

10. Wygląd naszego ekranu zdefiniujemy w metodzie `paint()`, zatem uzupełnij jej treść następująco:

```
//Malowanie tła
g.setColor(247, 223, 62);//zmiana aktualnego koloru
//Namalowanie wypełnionego aktualnym kolorem prostokąta
g.fillRect(0, 0, szerokośćEkranu, wysokośćEkranu);
//Wyświetlenie współrzędnych lewego górnego narożnika ekranu
g.setColor(0, 0, 0);
g.drawString("(0,0)", 0, 0, g.TOP | g.LEFT);
//Wyświetlenie współrzędnych prawego dolnego narożnika ekranu
g.drawString("(" + szerokośćEkranu + "," + wysokośćEkranu +
     ")", szerokośćEkranu, wysokośćEkranu, g.BOTTOM | g.RIGHT);
//Tekst na środku ekranu
g.setColor(0,0,255);
g.drawString(etykieta, szerokośćEkranu / 2, wysokośćEkranu / 2,
                  g.HCENTER | g.TOP);
//Wycinek elipsy
g.setColor(255, 0, 0);
g.fillArc(szerokośćEkranu/2-szerokośćElipsy/2, wysokośćEkranu/2-szerokośćElipsy/2,
          szerokośćElipsy, szerokośćElipsy / 2, 20, 220);
```



Zbudowaliśmy klasę wewnętrzną `EkranG` dziedziczącą z `Canvas`.

11. Aby zobaczyć efekty naszej pracy, musimy stworzyć logiczną reprezentację ekranu. Wykonaj tą czynność samodzielnie (patrz. pkt.III.2).
12. W publicznej części klasy `MIDletKlasyCanvas` zadeklaruj zmienną, która będzie wskazywać na obiekty klasy `EkranG` następująco: `EkranG ekranG;`

    Natomiast w konstruktorze tej klasy dopisz instrukcję (konstruktor powinien zostać utworzony po wykonaniu pkt. VI.11).

```
ekranG = new EkranG("J2ME rządzi");
```

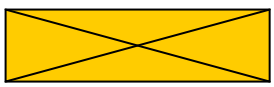
13. Uzupełnij treść metody `startApp()` następująco:

```
display.setCurrent(ekranG);
```

14. Uruchom program <**F6**>.
15. Zmodyfikuj samodzielnie tak program, aby zamknięcie aplikacji nastąpiło po wybraniu opcji z napisem **KONIEC.** (*Zastanów się gdzie umieścić odpowiednie instrukcje*).

VII Zadanie do samodzielnego rozwiązania

1. Utwórz nowy projekt typu **Mobile Application** o nazwie **J2ME-Trzeci** na dysku G:\ Nadaj nazwie MIDletu nazwę **Koperta**.
2. Zbuduj aplikację, która narysuje na ekranie telefonu komórkowego kopertę o wymiarach 120x40 umieszczoną po środku ekranu.
3. Wypełnij tło ekranu oraz koperty dowolnym kolorem. Linie koperty wyrysuj kolorem czarnym.



4. Poniżej koperty umieść *swoje nazwisko i imię* wyświetlone kolorem niebieskim.
5. Zamknięcie aplikacji powinno nastąpić po wybraniu opcji z napisem WYJŚCIE