4. Grafika

Rane verzije Jave omogućavale su veoma limitirane grafičke sposobnosti. Java 2 je omogućila mnogo bogatiji skup grafičkih operacija. Od tog skupa koji obuhvaća crtanja počevši od najjednostavnijih linija, preko drugih geometrijskih oblika pa sve do manipulacije s fotografijama u ovome poglavlju preći ćemo maji dio.

Sadržaj:

1. Korištenje grafike.
2. Applet koji crta linije.
3. Drugi oblici.
4. Boje.
5. Bojanje likova.
6. Pisanje teksta.
7. Smajli.
8. Čitanje ulaza.
9. Daljnja čitanja.
10. Zadaci

---

1. Korištenje grafike

Grafika je tema koju je zgodno uključiti rano unutar predavanja o programiranju jer programeri vole raditi s grafikom, a Java klase koje se odnose na grafiku daju dobar primjer mnogih osobina Jave koje ćemo i dalje susretati na ovim predavanjima. Cijena koju treba platiti je da je potrebno uvest drugačiji način pokretanja programa nego što je to za programe koji nisu koristili grafiku.

Postoje dva načina na koji se mogu pokrenuti Java programi: **aplikacije** i **appleti**. Svi programi koje smo dosada analizirali bili su aplikacije. To znači da su bili pokretani od strane Java SDK interpretera (java) i izvršavanje je započinjalo unutar `main` metode.
Sve aplikacije koje pišemo unutar ovih predavanja ispisivale su u prozor konzole (DOS prozor) koji je prikladan samo za ispisivanje karaktera. Ako bi aplikacija trebala proizvesti neku grafiku bilo bi potrebno da kreira grafički prozor. To je u Javi moguće, ali nećemo se time baviti u ovim predavanjima.

Applet je Java program koji se izvršava unutar Web pretraživača, poput Internet Explorera , ili Netscape pretraživača. Kada Web pretraživač stavlja sadržaj stranice na ekran i ako ta stranica sadrži link na applet, pretraživač će odvojiti dio stranice odnosno područje prikaza (display area) gdje će applet moći prikazati bilo kakvu informaciju, uključujući i grafiku.

Nakon toga će pretraživač započeti s izvršavanjem appleta. To će moći jer sadržava vlastiti Java interpreter.

Kraće rečeno: napišite applet koji može iscrtavati grafiku, napišite Web stranicu koja sadrži link na istu i zatražite od pretraživača da prikaže Web stranicu.

Postoji određeni problem s ovim pristupom. Pojedini pretraživači ne izvršavaju Java 2 programe.
Bilo da slabo prate napredovanje Jave ili namjerno ne podržavaju Javu. Dva su načina da riješimo taj problem:

Rješenje 1. Nabavite *plug-ni* za pretraživač koji će omogućiti pokretanja Java 2 programa.

Rješenje 2. Koristite **appletviewer**. Ovaj program je dio SDK. Pokrenut će bilo koji applet.

Appletviewer ima prvenstvenu namjenu testiranja funkcionalnosti appleta, ali predstavlja najjednostavniji način da vidimo što naš grafički program iscrtava.

Dakle, kada želimo kreirati neki grafički ispis, napisat ćemo applet koji crta grafiku u područje prikaza, zatim napisati malu Web stranicu koja samo sadržava link na applet. Zatim ćemo iskoristiti appletviewer za procesiranje Web stranice.

Jedini nedostatak ovoga pristupa je da appletviewer ne prikazuje ništa osim onoga što proizvede applet. dakle nećemo se baviti kreacijom Web stranica.

Sve što je potrebno znati o HTML za potrebe predavanja je slijedeće.
Web stranica sastoji se od tekst datoteke koja koristi HTML notaciju. U toj datoteci bit će samo link na datoteku Prog.class u kojoj će biti prevedeni Java applet kojega će izvršiti.

```
<APPLET CODE="Prog.class" WIDTH=300 HEIGHT=300>
</APPLET>
```

Osim linka navedena je i veličina područja prikaza koje će biti dodijeljeno appletu na korištenje:

```
WIDTH=300 HEIGHT=300
```

---

2. Applet koji crta linije

Applet je također jedna Java klasa. U mnogim pogledima je nalik na klase koje smo koristili dosada. Sastavljen je od polja, metoda i konstruktora. Međutim razlikuje se od klasa koje smo dosada koristili u jednoj bitnoj stvari. Posjeduje najmanje jedna metoda koja je namijenjena da je koristi *windows manager* . Windows manager je objekt koji održava prozore na ekranu, omogućavajući njihovo otvaranje, skrivanje(minimiziranje), zatvaranje , itd..

Jedna od metoda koje ćemo priskrbiti za windows manager je metoda *paint*. Windows manager će prvi put pozvati tu metodu kad se područje prikaza pojavi na ekranu. Također kad god bude bilo potrebno osvježiti sadržaj ekrana bit će pozvana metoda paint. Osvježavanje prikaza bit će potrebno npr. kad pomicanjem(skroliranjem) Web stranice područje prikaza izađe izvan vidljivosti pa se novim pomicanjem opet vrati na ekran. U tom slučaju sadržaj područja prikaza mora se ponovo iscrtati odnosno osvježiti. Drugi primjer kad je to potrebno je kad preko područja prikaza npr. otvorimo prozor neke druge aplikacije. Kada zatvorimo taj prozor opet će biti potrebno ponovo osvježiti sadržaj koji je dotada bio pokriven prozorom.

Zadatak `paint` metode je da isporuči (render) sadržaj u područje prikaza (nacrta linije, oboji područja, napiše tekst ,...). Da bi to napravili moramo koristiti jedan objekt koji nazivamo **Graphics2D** objekt. To je nešto slično kao `System.out` objekt koji smo koristili za ispis

karaktera u prozoru konzole. Međutim Graphics2D objekt spojen je na područje prikaza appleta (visoka rezolucija), a ne primitivni DOS prozor.

Graphics2D objekt posjeduje niz metoda kao npr. `drawString` kojim možemo ispisivati tekst, metodu `draw` koja može iscrtavati oblike poput kruga, kvadrata, metodu `drawImage` koju možemo koristiti za ispis kompletnih slika. Također Graphics2D objekt posjeduje metode kojima se može zadati određena boja ili uzorak kojom će se crtati neki lik ili ispisivati tekst. Posjeduje i metode kojima se može zadati neka grafička transformacija lika npr. rotacija.

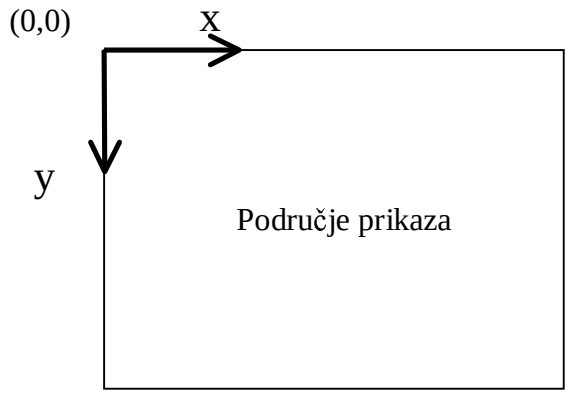
Graphics2D objekt ne moraju biti spojeni samo na područje prikaza na ekranu. Možemo ih spojiti npr. na printer ili na neko područje u memoriji (slika će biti pohranjena u memoriju).

Pretpostavimo da je `g2` objekt tipa Graphics2D spojen na područje prikaza appleta. Pretpostavimo da je `line1` linija između dvije točke. Slijedeća naredba će prikazati liniju na ekranu:

```
g2.draw(line1);
```

Postoje različiti geometrijski oblici koji mogu biti iscrtani ili obojani korištenjem Graphics2D objekta. Koristit ćemo 4 tipa: linije, pravokutnike (rectangles), elipse (uključujući i krugove) te dijelove elipsi tj. lukove (arcs). Svaki od ovih geometrijskih oblika u Javi je reprezentiran kao klasa. Npr. linija je reprezentirana objektom tipa `Line2D.Double`. Riječ `Double` pokazuje da se podaci o liniji pohranjeni ka 64-bitni broj s pokretnim zarezom.(Ako želite uštediti memoriju na raspolaganju je i `Line2D.Float`).

Da bismo kreirali liniju potrebno je specificirati koordinate njenih krajnjih točaka. Pozicije su dane ka x i y koordinate i njihova orijentacija je slijedeća:



Koordinate se mjere u pikselima (pixel = picture element). Npr. područje koje ima 300×300 piksela protezat će se otprilike na trećini ekrana.

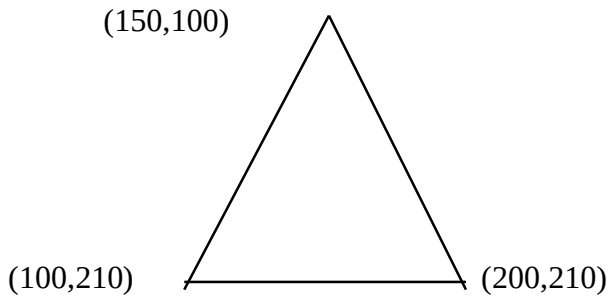
da bismo kreirali objekt linije koristit ćemo `Line2D.Double` konstruktor koji ima koordinate krajnjih točaka linije kao parametre:

```
Line2D.Double(x0,y0,x1,y1)
```

Kreira objekt koji predstavlja liniju od (x0,y0) do (x1,y1).
(Vrijednosti x0, y0, x1 i y1 su `double`. Ako napišete cjelobrojne koordinate Java će ih konvertirati u `double`.)

Primijetite da kreiranje Line2D.Double objekta **nije** isto što i njegovo prikazivanje na ekranu. Taj objekt bit će prikazan na ekranu tek onda kad ga proslijedimo `draw` metodi Graphics2D objekta koja će ga iscrtati u području prikaza.

Npr. želimo nacrtati slijedeći trokut na ekranu:



Bit će potrebno koristiti slijedeće izraze. Prvo kreiramo objekt koji predstavlja lijevu stranu trokuta tj. objekt tipa Line.Double te ga dodijelimo varijabli `line1.` Zatim ga iscrtamo korištenjem `g2.draw` metode. proces ponovimo za ostale linije.

```
Line2D.Double line1 =
   new Line2D.Double(150,100,200,210);
g2.draw(line1);

Line2D.Double line2 =
   new Line2D.Double(200,210,100,210);
g2.draw(line2);

Line2D.Double line3 =
   new Line2D.Double(100,210,150,100);
g2.draw(line3);
```

Pretpostavimo da pišemo applet koji će samo crtati taj trokut. Napisat ćemo `paint` metodu koja koristi prije navedene naredbe. Postavlja se pitanje kako ćemo kreirati Graphics2D objekt tj. označen s `g2 ?` Odgovor je da ga nećemo mi kreirati. Windows manager će kreirati taj objekt i proslijediti ga `paint` metodi kao parametar.

Postoji jedan mali tehnički problem na koji u svemu moramo paziti. Appleti koriste paint metode još iz najranijih dana Jave, davno prije nego što je Java 2 uvela Graphics2D objekte. Prije je parametar `paint` metoda imala parametar koji je bio objekt tipa *Graphics* (bez 2D !). Taj objekt je bio sličan Graphics2D objektu, ali daleko manje funkcionalnosti.
Problem je u tome što Java 2 treba pokretati i stare programe pisane za ranije verzije Jave. Zato je ostavljeno da Java kao parametar `paint` metode appleta očekuje Graphics objekt što ćemo morati napisati u zaglavlju paint metode:

```
public void paint(Graphics g)
```

Prevodioc se neće buniti što windows će manager proslijediti Graphics2D objekt kao parametar kad bude pozivao paint metodu ! To je zato što je Graphics2D objekt specijalni slučaj Graphics objekta.

Međutim prevodioc će se pobuniti ako počnemo objekt označen s g tretirati kao Graphics2D objekt. Morat ćemo napisati slijedeći izraz:

```
Graphics2D g2 = (Graphics2D) g;
```

Ovaj izraz kaže: kreiraj Graphics2D varijablu, `g2`, i dodijeli joj objekt g *za kojeg obećavamo da je Graphics2D objekt*. Kastiranje `(Graphics2D)` je naše obećanje prevodiocu da je parametar stvarno Graphics2D objekt. U ostatku metoda g2 tretiramo kao Graphics2D objekt.

Slijedi kompletan program koji će iscrtati trokut. Potrebna je samo jedna klasa. Zaglavlje klase sadrži riječi `extends applet`. Tim pokazujemo prevodiocu da ta klasa definira applet. Prve tri linije pokazuju Java prevodiocu koje tri linije Java biblioteke će biti potrebne.

PRIMJER 1

```
import java.applet.*;
import java.awt.*;
import java.awt.geom.*;

/* Applet koji prikazuje trokut . */

public class Triangle extends Applet

{  public void paint(Graphics g)
   {  Graphics2D g2 = (Graphics2D) g;

      Line2D.Double line1 =
         new Line2D.Double(150,100,200,210);
      g2.draw(line1);

      Line2D.Double line2 =
         new Line2D.Double(200,210,100,210);
      g2.draw(line2);

      Line2D.Double line3 =
         new Line2D.Double(100,210,150,100);
      g2.draw(line3);
   }
}
```

Primijetite da bi se neke naredbe mogle sažetije pisati ako bismo izbjegli uvođenje pomoćnih varijabli `line1, line2 , line3.`

Umjesto pisanja:

```
Line2D.Double line1 =
   new Line2D.Double(150,100,200,210);
g2.draw(line1);
```

Mogli smo pisati

```
g2.draw(new Line2D.Double(150,100,200,210));
```

Obje verzije su ispravne.

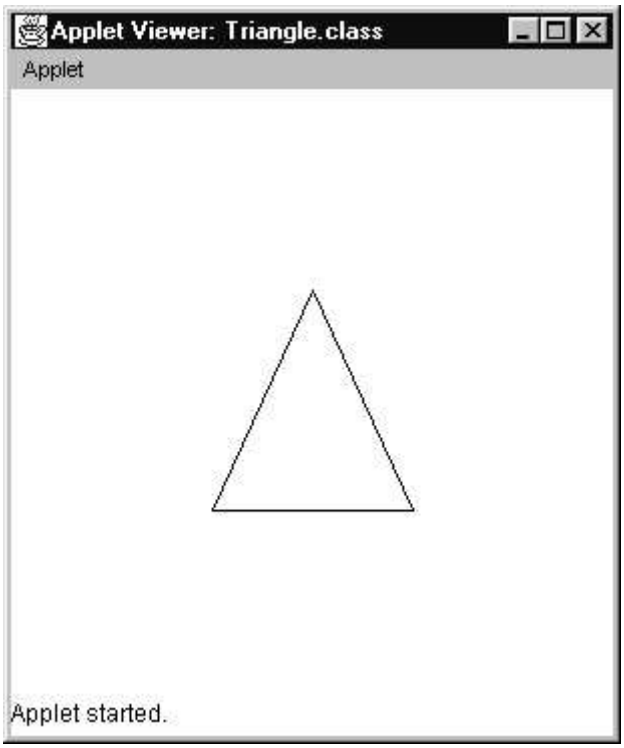
Da bismo pokrenuli ovaj program može biti korištena slijedeća tekstualna datoteka. Ona sadrži ključnu vezu s prevedenom Triangle klasom.

```
<APPLET CODE="Triangle.class"
        WIDTH=300
        HEIGHT=300>
</APPLET>
```

(Smještaj linija nije bitan. Možete otkucati sve u jednoj liniji ako tako želite). Pretpostavimo da je ta datoteka nazvana `Triangle.txt`. Nakon što smo preveli Triangle.java utipkajmo:

```
appletviewer Triangle.txt
```

i slijedeći prozor će se pojaviti na ekranu.



Upamtite. Trokut se iscrtao zato što windows manager poziva `paint` metodu. Pretpostavim da pokušate pokvariti sadržaj prozora tako da npr. pokrenete neki drugi program koji će

prekriti applet prozor. Čim učinite nešto što će vratiti vidljivost appleta windows manager će odmah ponovo pozvati `paint` metodu i ponovo iscrtati sadržaj prozora.

Java biblioteka je podijeljena u veći broj dijelova koji se nazivaju **paketi** ili **packages.**
Prva linija primjera:

```
import java.applet.*;
```

kaže prevodiocu da će se koristiti paket koji se naziva `java.applet` . Oznaka `*` pokazuje prevodiocu da očekuje upotrebu *bilo koje* klase iz navedenog paketa. U primjeru se koristi samo jedna klasa iz `java.applet` paketa. Da nismo *importirali* `java.applet` klasu bili bismo prisiljeni koristiti puno ime `java.applet.Applet` umjesto samo `Applet`. To je jedini razlog korištenja `import` naredbe.

U `java.awt` paketu sadržane su mnoge grafičke klase. U prethodnom primjeru koristili smo `Graphics` i `Graphics2D` klase. Npr. paket `java.awt.geom` omogućava upotrebu geometrijskih oblika poput `Line2D.Double` objekta.

Nije potrebno importirati sve pakete Java biblioteke. Paket `java.lang` uključen je automatski. On uključuje veliki broj često korištenih klasa, poput klasa. `String`, `Math` i `System`.)

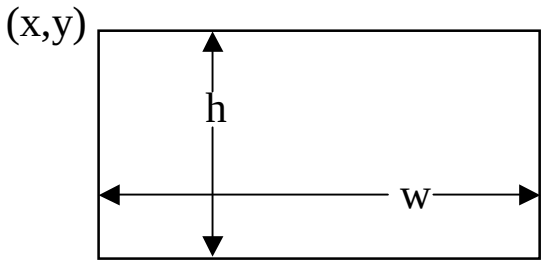
---

3. Drugi oblici

nova grafička biblioteka u javi 2 uključuje različite korisne grafičke oblike. Jedan od njih je i pravokutnik, sadržan u klasi `Rectangle2D.Double`. Pravokutnik možemo konstruirati pomoću slijedećeg konstruktora :

```
Rectangle2D.Double(x,y,w,h)
```

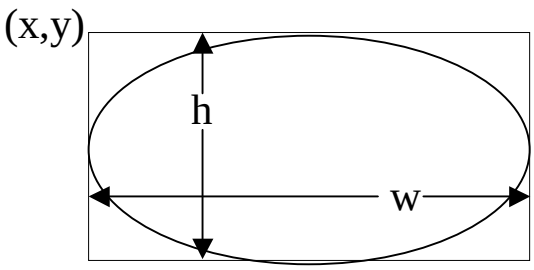
Gornji lijevi kut pravokutnika ima koordinate (x,y). Njegova širina je w, a visina je h.



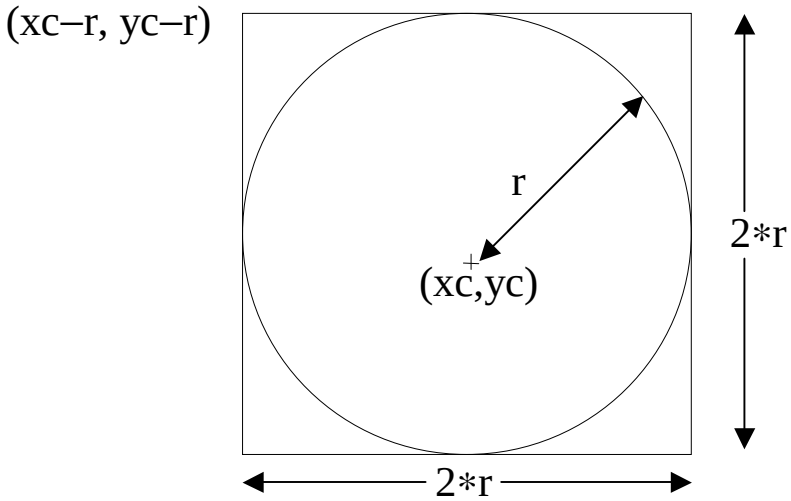
Slično vrijedi i za elipsu koja je definirana klasom Ellipse2D.Double. Može biti kreirana slijedećim konstruktorom

```
Ellipse2D.Double(x,y,w,h)
```

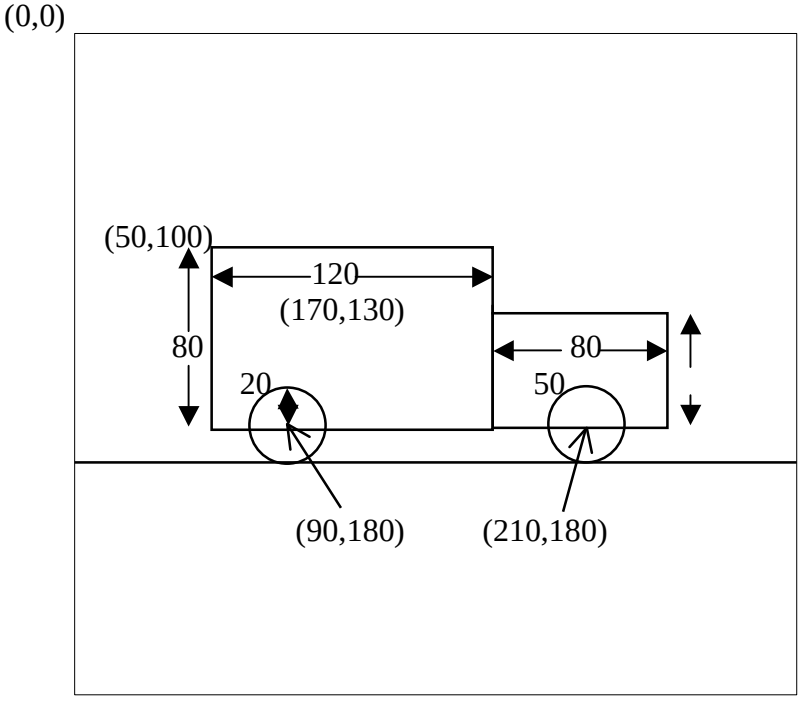
Pozicija i oblik elipse definiran je parametrima x, y, w i h. Ovi parametri određuju pravokutnik koji sadržava elipsu i zadaju se isto kao i za pravokutnik.



Ne postoji objekt koji bi bio rezerviran za definiciju kruga. međutim krug možemo nacrtati kao poseban slučaj elipse čija je širina jednaka visini. Ako je centar kruga u (xc,yc), a njegov radijus je r, tada je to ekvivalentno elipsi koju definiramo preko pravokutnika sa gornjim lijevim kutom u (xc−r, yc−r) te širinom 2∗r i visinom 2∗r.



U slijedećem primjeri koristit ćemo linije, pravokutnike i elipse te nacrtati jedan mali kamion. Prije nego počnemo pisati kod potrebno je pažljivo definirati položaj pojedinih likova. Slijedi skica kamiona u području prikaza od 300×300.

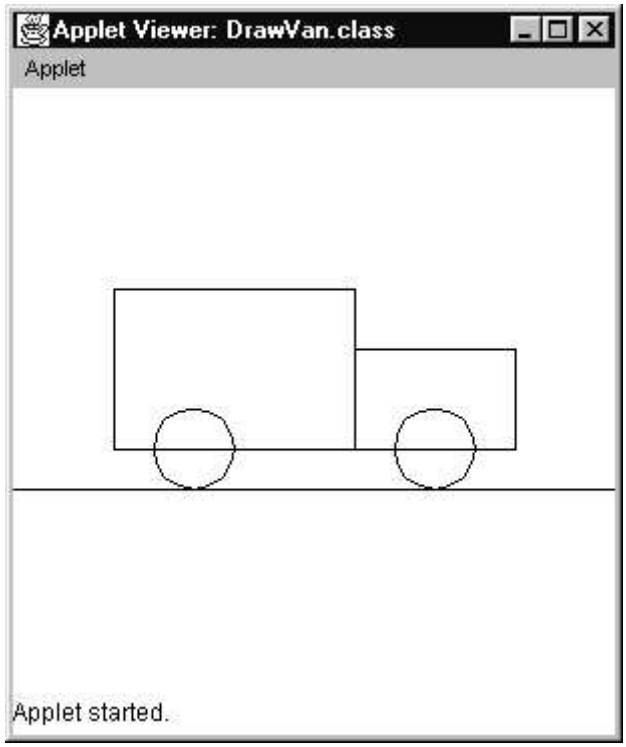


Slijedi applet koji će nacrtati ovu sliku. Za njega je kamion predstavljen kao dva pravokutnika plus dva kruga. Tu je još i linija ceste.

PRIMJER 2

```
import java.applet.*;
import java.awt.*;
import java.awt.geom.*;

public class DrawVan extends Applet
{  public void paint(Graphics g)
   {  Graphics2D g2 = (Graphics2D) g;

      Rectangle2D.Double back =
         new Rectangle2D.Double(50,100,120,80);
      g2.draw(back);

      Rectangle2D.Double back =
         Rectangle2D.Double(170,130,80,50);
      g2.draw(back);

      Ellipse2D.Double backWheel =
         new Ellipse2D.Double(70,160,40,40);
      g2.draw(backWheel);

      Ellipse2D.Double frontWheel =
         new Ellipse2D.Double(190,160,40,40));
      g2.draw(frontWheel);

      Line2D.Double baseLine =
         new Line2D.Double(0,200,300,200);
      g2.draw(baseLine);
   }
}
```

Slijedi prikaz kako bi se trebao vidjeti na ekranu:



4. Boja

Ako drugačije ne navedemo Graphics2D objekt će crtati crnom bojom. Međutim moguće je da promijenite boju kad god to zaželite. Klasa Graphics2D posjeduje `setColor` metodu koja će promijeniti boju u onu koju joj proslijedite kao parametar.

Boje su u Javi predstavljeni kao objekti tipa **Color**. Klasa Color sadrži trinaest konstanti koje predstavljaju standardne boje. (konstanta je varijabla s fiksnom vrijednošću). Npr. me]u njima je konstanta `Colour.red` koja predstavlja crvenu boju. Kompletna lista boja navedena je kasnije.

Stoga, ako želite da objekt g2 (klase Graphics2D) počne pisati crvenom bojom, onda upotrijebite slijedeći izraz prije korištenja `draw` metode:

```
g2.setColor(Color.red);
```

Od momenta kad je izvršena pa sve dok je ponovo ne izvršimo s drugom bojom kao argumentom, sve što `g2` bude prikazivao bit će obojeno crveno.

Objekt Color može predstavljati mnogo više boja nego standardnih 13. Možete definirati svoje boje specificiranjem količine tri osnovne boje : crvene, zelene i plave (red-green-blue = RGB).

Koristi se *aditivni* sustav boja. njegovo osnovno svojstvo je da ako kombiniramo maksimalnu količinu crvene, zelene i plave boje dobivamo bijelu boju. Kombiniranjem nultih vrijednosti crvene, zelene i plave boje dobivamo crnu boju (Ovo je suprotno korištenju boja u tiskanju ili slikarstvu. tu se koriste *substraktivni* sustavi. Npr. miješanjem različitih uljanih boja dobivate sve tamnije boje slične crnoj.

Klasa Color posjeduje konstruktor za kreiranje RGB boje

```
Color(r, g, b)
```

> Kreiraj boju kombinirajući r količinu crvene, g količinu zelene i b količinu plave. Ove vrijednosti su brojevi u pokretnom zarezu s vrijednostima od 0 do 1 tipa `float`.

Moramo pažljivo pisati te parametre. Ako npr. napišete 0.5 prevodioc će to shvatiti kao `double` vrijednost i javiti pogrešku. Potrebno je specificirati da se radi o tipu `float`, tj. napisati `0.5f`. Slijedi primjer definiranja žute boje:

```
Color c = new Color(1.0f, 1.0f, 0.0f);
```

(Ne pišite Color(1, 1, 0). Normalno možete koristiti cijele brojeve na mjestu gdje se očekuje float vrijednost, ali ne i ovdje ! Razlog je što Color klasa ima još jedan konstruktor s drukčijim parametrima koji su definirani kao cjelobrojni iznosi crvene, plave i zelene kao *cijeli brojevi u rasponu od 0 do 255 !* Ako dakle napišete Color(1, 1, 0), Java će pretpostaviti da koristite sasvim drugi konstruktor gdje ste specificirali neznatan iznos crvene i zelene i ništa plave te će rezultat biti skoro crna boja.
Slijedi lista 13 standardnih boja i količine crvene, zelene i plave od kojih se sastoje:

```
Color.black     0.0f        0.0f        0.0f
Color.blue      0.0f        0.0f        1.0f
Color.cyan      0.0f        1.0f        1.0f
Color.gray      0.5f        0.5f        0.5f
Color.darkGray  0.25f       0.25f       0.25f
Color.lightGray 0.75f       0.75f       0.75f
Color.green     0.0f        1.0f        0.0f
Color.magenta   1.0f        0.0f        1.0f
Color.orange    1.0f        0.8f        0.0f
Color.pink      1.0f        0.7f        0.7f
Color.yellow    1.0f        1.0f        0.0f
Color.white     1.0f        1.0f        1.0f
```

5. Bojanje likova

Objekt Graphics2D može se koristiti za bojanje unutrašnjosti likova poput pravokutnika ili elipse. Bit će upotrijebljena boja koja je zadnja postavljena s metodom `setColor`. Ako nijedna boja nije postavljena bit će upotrijebljena crna boja.
Metoda koju ćemo koristiti naziva se `fill`. Pozivamo je na slijedeći način:

```
g2.fill(s)
```

Koristi Graphics2D objekt referenciran s `g2` za prikaz oblika `s` ispunjenog s trenutnom bojom pridruženoj `g2`.

Npr. slijedeće naredbe kreirat će mali krug i prikazat ga ispunjenog žutom bojom.

```
Ellipse2D.Double sun =
   new Ellipse2D.Double(140,50,20,20);
g2.setColor(Color.yellow);
g2.fill(sun);
```

Slijedeći primjer koristi istu tehniku za crtanje kamiona sličnog kao u primjeru 1, ali obojenog u crvenu boju. Točkovi su bijeli, a gume su tamno sive boje. Linija ceste zamijenjena blijedo-plavom bojom. Primijetite da je potrebno iscrtati pozadinu prije kamiona.

PRIMJER 3

```
import java.applet.*;
import java.awt.*;
import java.awt.geom.*;


public class PaintVan extends Applet

{  public void paint(Graphics g)
   {  Graphics2D g2 = (Graphics2D) g;

      /* Paint the background. */
      Color paleBlue =
```

```
            new Color(0.75f, 0.750f, 1.0f);
         g2.setColor(paleBlue);
         g2.fill(new Rectangle2D.Double(0,0,300,300));

         /* Paint the body of the van. */
         g2.setColor(Color.red);
         g2.fill
            (new Rectangle2D.Double(50,100,120,80));
         g2.fill
            (new Rectangle2D.Double(170,130,80,50));

         /* Paint the back wheel. */
         g2.setColor(Color.darkGray);
         g2.fill(new Ellipse2D.Double(70,160,40,40));
         g2.setColor(Color.white);
         g2.fill(new Ellipse2D.Double(80,170,20,20));

         /* Paint the front wheel. */
         g2.setColor(Color.darkGray);
         g2.fill(new Ellipse2D.Double(190,160,40,40));
         g2.setColor(Color.white);
         g2.fill(new Ellipse2D.Double(200,170,20,20));
      }
   }
```

Slijedi obojani kamion:

6. Pisanje teksta

Osim za crtanje i bojanje oblika, Graphics2D objekt može se koristiti za ispis teksta na ekranu. Prije nego što to pokušamo učiniti, potrebno je Graphics2D objektu naznačiti koji će *font* koristiti. Font je reprezentiran s objektom tipa `Font` . Koristimo slijedeći konstruktor za kreiranje Font objekta.

`Font(familija, stil, veličina)` (family, style, size)

Parametar `familija` je string koji predstavlja naziv skupa srodnih fontova. Primjeri srodnih familija su Times Roman koji se koristi kao glavni font ovog teksta, i Courier koji se koristi za tekst programa. Ova rečenica je tiskana u Arial fontu. Slijedeća logički nazivi mogu se koristiti za familije fontova:

```
Serif
SansSerif
Monospaced
Dialog
DialogueInput
```

Parametar `stil` ima jednu od slijedećih vrijednosti koje naznačuju da li se koristi *italic (zakošeno) ili* **bold (podebljano)** pisanje:

```
Font.PLAIN
Font.ITALIC
Font.BOLD
Font.ITALIC+Font.BOLD.
```

Parametar `veličina` je *veličina fonta u točkama*. To je visina karaktera mjerena u jedinci od 1/72 inča. Veličina točke od 1 približno odgovara jednom pikselu. Ova predavanja napisana su fontom veličine 12 i karakteri bi trebali biti visoki oko 1/6 inča.

Jednom kada kreirate Font objekt, potrebno je Graphics2D objektu narediti da ga koristi. To činimo upotrebom slijedeće metode:

`g2.setFont(f)`
Postavi Graphics2D objekt referenciran s `g2` tako da koristi `f` za pisanje teksta.

Sve što preostaje je da se ispiše string na određeno mjesto na ekranu:

`g2.drawString(text, x, y)`

Koristi Graphics2D objekt `g2` za ispis stringa `text`, počevši od pozicije (x,y) u području prikaza. x i y su of tipa `int`. (x je pozicija lijeve strane teksta, a y je visina na kojoj se nalazi bazna linija teksta.

Slijedeći primjer ilustrira upotrebu logičkih familija fontova . U svakom koraku Graphics2D objektu pridružujemo jedan logički font te pomoću tog fonta ispisujemo naziv familije. U svakom od slučajeva koristimo stil FONT.PLAIN te je veličina fonta jednaka 30. 0.

PRIMJER 4

```
import java.applet.*;
import java.awt.*;

public class Fonts extends Applet
{  final int SIZE = 30;

   public void paint(Graphics g)
   {  Graphics2D g2 = (Graphics2D) g;

      Font font1 =
         new Font("Serif", Font.PLAIN, SIZE);
      g2.setFont(font1);
      g2.drawString("Serif", 50, 50);

      Font font2 =
         new Font("SansSerif", Font.PLAIN, SIZE);
      g2.setFont(font2);
      g2.drawString("SansSerif", 50, 100);

      Font font3 =
         new Font("Monospaced", Font.PLAIN, SIZE);
      g2.setFont(font3);
      g2.drawString("Monospaced", 50, 150);

      Font font4 =
         new Font("Dialog", Font.PLAIN, SIZE);
      g2.setFont(font4);
      g2.drawString("Dialog", 50, 200);

      Font font5 =
         new Font("DialogInput", Font.PLAIN, SIZE);
      g2.setFont(font5);
      g2.drawString("DialogInput", 50, 250);
   }
}
```

Za slijedeći primjer korištenja ispisa teksta u primjeru 3 (koji crta obojani kamion) dodajte slijedećih pet naredbi na kraj `paint` metode.

```
/* nacrtaj logo na stranici kamiona */
Font f = new Font("Serif", Font.ITALIC, 25);
g2.setFont(f);
g2.setColor(Color.white);
g2.drawString("Java",66,124);
g2.drawString("Delivers",66,148);
```

7. Smajli

Grafički objekti koje smo koristili u ovom poglavlju, poput Line2D.Double objekta na neki način posjeduju pridruženi prikaz. Ovu ideju možemo dalje širiti. Ako bismo trebali napraviti grafički program za očekivat bi bilo da imamo niz Java objekata za koje bi trebalo realizirati prikaz na ekranu. U slijedećem primjeru ćemo napraviti jedan takav objekt.

Definirat ćemo tip objekta koji predstavlja osobu. Nazvat ćemo je Person (osoba) klasa. Postoje mnogi atributi koje bismo mogli definirati za objekt tipa Person. U ovom slučaju upotrijebit ćemo samo jedan atribut koji će definirati raspoloženje osobe.
To je vrijednost između 0 i 1 i nazvat ćemo je `raspoloženje`. Osoba s vrijednošću `raspoloženje` ode 1 je veoma raspoložena osoba. 0 znači da je u pitanju potpuno neraspoložena osoba.

Jedna od operacija koje ćemo asocirati s objektom Person je crtanje lika osobe. To će raditi slijedeća metoda:

```
p.crtajLice(double x, double y, Graphics2D g2)
```

Nacrtaj sliku lica osobe reprezentirane s Person objektom referenciranog s varijablom p. Centar lica su koordinate `x` i `y`. Slika će se iscrtati pomoću Graphics2D objekta `g2`.

Lice će biti jednostavno sa osmjehom proporcionalnim raspoloženju.

Imat ćemo dva metoda koja će uticati na raspoloženje:

`p.oraspoloži()` // Povećaj raspoloženje osobe za konstantan faktor.

`p.oneraspoloži()` // Smanji raspoloženje osobe za konstantan faktor.

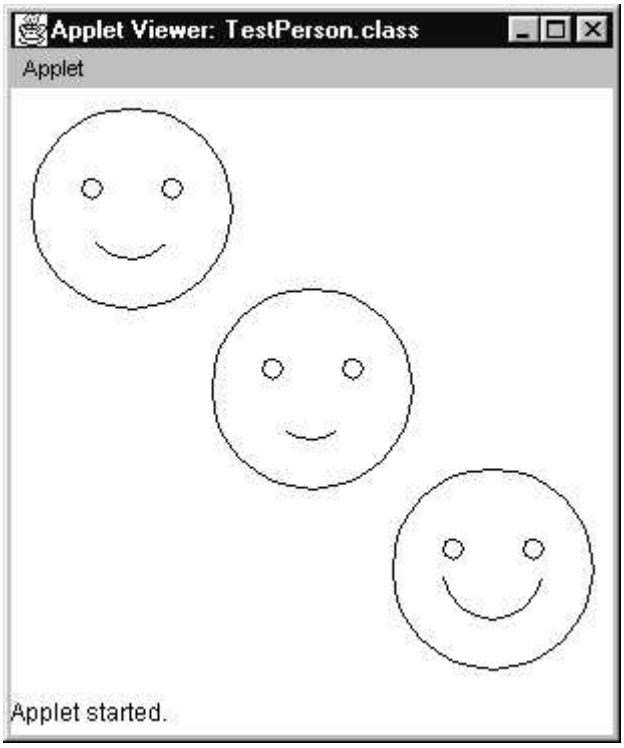
Konstruktor za kreiranje Person objekta:

`Person(double h)` // Kreiraj Person objekt s 'raspoloženje' postavljenim na `h`.

Prije same implementacije Person klase, slijedi `paint` metoda koji kreira Person objekt i mijenja mu raspoloženje nekoliko puta. Ova metoda crta lice osobe na tri različite lokacije u području prikaza appleta.

PRIMJER 5

```
public void paint(Graphics g)
{  Graphics2D g2 = (Graphics2D) g;

   Person kim = new Person(0.5);
   kim.crtajLice(60,60,g2);
   kim.oneraspoloži();
   kim.crtajLice(150,150,g2);
   kim.oraspoloži();
   kim.oraspoloži();
   kim.oraspoloži();
   kim.oraspoloži();
   kim.crtajLice(240,240,g2);
}
```



Sada slijedi implementacija. Kako pretpostavljate , klasa Person sadržava jedno polje nazvano `raspoloženje`. Polje je tipa `double`, i sadrži vrijednost u opsegu od 0 do 1. Konstruktor postavlja to polje na vrijednost parametra.

Metoda `oneraspoloži` množi vrijednost raspoloženja s konstantnim faktorom. Taj faktor se drži u varijabli nazvanoj `factor`. Tijelo metode sadrži samo jednu naredbu:

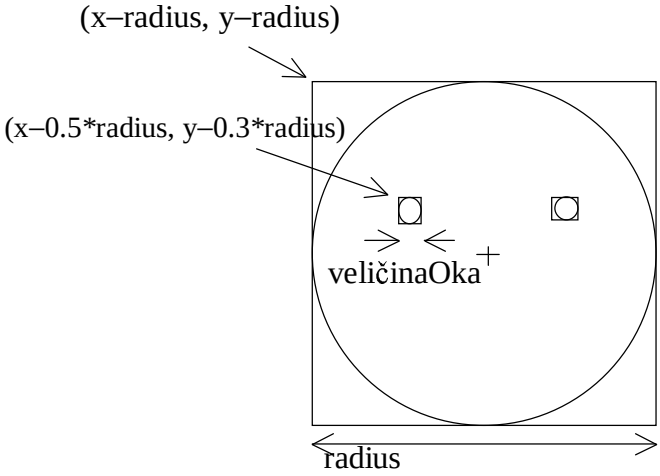
```
raspoloženje = factor*raspoloženje;
```

`factor` je postavljen unaprijed na 0.65. Bilo je jednostavnije pisati `raspoloženje = 0.65*raspoloženje`. Umjesto toga smo ipak smo definirali konstantu i dali joj naziv 'factor'. Poslije ćemo lakše promijeniti njenu vrijednost na jednom mjestu nego na svim mjestima gdje je budemo koristili.

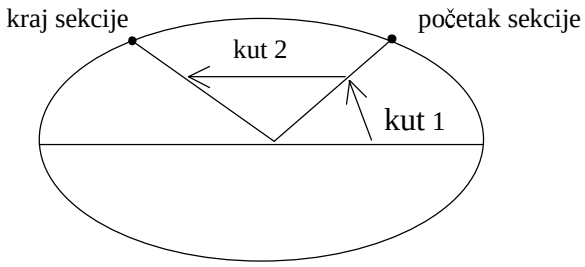
Metoda `oraspoloži` izvodi slijedeći izraz:

```
raspoloženje = 1 - factor*(1 - raspoloženje);
```

Jedini 'teži' dio implementacije klase Person je implementacija metoda `crtajLice`. Svako lice sastojat će se od konture, lijevog oka desnog oka i osmjeha. Prva tri su Ellipse2D.Double objekti. Parametri korišteni u konstrukciji lica su izračunati koristeći vrijednosti `x` i `y` koji predstavljaju centar lica., `radius` koji je radijus lica, `veličinaOka` koji je dijametar svakog oka.



Konstrukcija osmjeha je kompliciranija jer moramo koristiti novi objekt iz java biblioteke tj. Arc2D.Double objekt. Ovaj objekt predstavlja dio elipse – sekciju. Potrebno je specificirati kut koji predstavlja početak sekcije te kut koji predstavlja kut sekcije.



Postoji više konstruktora koji su mogući za Arc2D Double objekt. Slijedi jedan od njih:

```
Arc2D.Double(double x, double y,
             double w, double h,
             double start, double extent,
             int type)
```

Kreiraj sekciju elipse s gornjim lijevim kutom na (x,y), širine w i visine h. Sekcija je definiran početnim kutom start i drugim kutom extent. Vrsta sekcije definirana je parametrom type.

Za kreirati samo dio sekcije koji se sastoji od krivulje (bez spojenih krajeva) parametar type treba postaviti na konstantu Arc2D.Double.OPEN.

Slijedi naredba za konstrukciju osmjeha:

```
Shape osmjeh =
   new Arc2D.Double
      (x-0.5*radius, y-0.5*radius, radius, radius,
       180+90*(1-raspoloženje), 180*raspoloženje,
       Arc2D.Double.OPEN);
```

Ako pažljivije pogledate vidjet ćete da je osmjeh dio kruga s centrom u centru lica s duplo manjim radijusom od radijusa lica. Ako je raspoloženje jednako 1, sekcija je polukrug. Ako je raspoloženje 0 onda sekcija u potpunosti nestaje.

Varijabla osmjeh je deklarirana kao klasa tipa Shape, ne kao Arc2D.Double. U stvari svaki geometrijski oblik koji se može nacrtati pomoću objekta Graphics može biti tipa Shape. 'Shape' je primjer Java sučelja (**interface**). Interface Shape definira koje metode neki objekt treba posjedovati da bi se mogao referirati varijablom tog tipa.

Slijedi kompletna definicija klase Person

PRIMJER 5b

```
import java.awt.*;
import java.awt.geom.*;

/* Person objekt predstavlja osobu
   s različitim nivoom raspoloženja raspoloženje.
*/

public class Person
{
   prihvate double raspoloženje;

   prihvate double factor = 0.65;
```

```
      // Faktor korišten za povećanje ili smanjenje
      // raspoloženja osobe

   /* Oraspoloži osobu
   */
   public void oraspoloži()
   {  raspoloženje = 1 - factor*(1 - raspoloženje);
   }

   /* Oneraspoloži osobu
   */

   public void oneraspoloži()
   {  raspoloženje = factor*raspoloženje;
   }

   /* Nacrtaj lice osobe korištenjem Graphics2D objekta g2.
      Centar lica je (x,y), radius 50.
   */
   public void
      crtajLice(double x, double y, Graphics2D g2)
   {  double radius = 50;
      double veličinaOka = 10;

      Shape outline =
         new Ellipse2D.Double
            (x-radius, y-radius, 2*radius, 2*radius);
      Shape leftEye =
         new Ellipse2D.Double
            (x-0.5*radius, y-0.3*radius,
             veličinaOka, veličinaOka);
      Shape rightEye =
         new Ellipse2D.Double
            (x+0.5*radius-veličinaOka, y-0.3*radius,
             veličinaOka, veličinaOka);
      Shape osmjeh =
         new Arc2D.Double
            (x-0.5*radius, y-0.5*radius,
             radius, radius,
             180+90*(1-raspoloženje), 180*raspoloženje,
               Arc2D.Double.OPEN);

      g2.draw(outline);
      g2.draw(leftEye);
      g2.draw(rightEye);
      g2.draw(osmjeh);
   }

   /* Kreiraj osobu tj. objekt tipa Person s raspoloženjem
    postavljenim na h   */
   public Person(double h)
   {  raspoloženje = h;
   }
}
```

8. Čitanje ulaza

U appletima moramo unošenju podataka s tipkovnice pristupiti na različit način nego što smo to radili u dosadašnjim aplikacijama.

Koristit ćemo se sekundarnim prozorima koje ćemo nazvati *dialog*.
Kreirat ćemo ih korištenjem **JOptionPane** objekta. Ova klasa realizira dialog koji sadrži poruku od korisnika te prostor za upis podataka.

Slijedeća rečenica će uzrokovati pojavu dijaloga s porukom 'Please enter your name.' Dijalog će čekati da korisnik upiše niz znakova i pritisne OK (ili pritisne Enter). Na kraju će uneseni string biti pridružen varijabli reply.

```
String reply =
   JOptionPane.showInputDialog
      (”Please enter your name.”);
```

To će izgledati ovako:

Istu tehniku možete koristiti za unos brojeva. Prvo učitajte korisnikov upis kao string, st, i onda ga konvertirajte korištenjem slijedećih metoda:

`Integer.parseInt(st)` ili `Double.parseDouble(st)`.

U primjeru 6 program opet crta smajlija. Ovaj put program preko dijaloga pita korisnika da unese broj koji odgovara raspoloženju osobe. Na osnovu unesenog broja konstruira se Person objekt s unesenim raspoloženjem.

Gdje ubaciti izraz za prikaz dijaloga za unos početnog raspoloženja. Nije dobro mjesto Paint metoda jer se ona izvršava svaki put kada windows manager osvježava prikaz na ekranu. Potrebno je da se dialog pojavi samo jednom prilikom pokretanja appleta. Pravo mjesto je metoda

```
public void init()
```

Osim `paint` metode windows manager poziva i metodu `init` ,ali samo jednom na početku izvršavanja appleta. Stoga se u tu metodu mogu smjestiti naredbe koje su vezane za inicijalizaciju parametara appleta.

Primijetite da je objekt tipa Person dodijeljen polju `fred` kojemu se kasnije pristupa u metodi `paint`. Nismo mogli deklarirati varijablu tipa Person unutar metode Init !

PRIMJER 6

```
import java.applet.*;
import java.awt.*;
import javax.swing.*;
   // (Potrebno za JOptionPane klasu.)

/* Čitaj 'raspoloženje' vrijednost i
   zatim nacrtaj lice.
*/

public class DrawSmiley extends Applet

{   Person fred;

    public void init()
    {  String input =
          JOptionPane.showInputDialog
             ("Koliko je osoba sretna?");
       double h = Double.parseDouble(input);
       fred = new Person(h);
    }

    public void paint(Graphics g)
    {  Graphics2D g2 = (Graphics2D) g;
       fred.crtajLice(150, 150, g2);
    }
}
```

Ovaj applet je napravljen za dimenzije područja prikaza od 300×300.
Možete applet učiniti pametnijim tako da prvo provjeri dimenzije trenutne dimenzije područja prikaza. Metoda `getWidth()` vratit će trenutnu širinu područja prikaza, dok će metoda `getHeight()` vratiti trenutnu visinu.

Dakle možete pisati slijedeće:

```
double x = getWidth()/2.0;
double y = getHeight()/2.0;
fred.crtajLice(x, y, g2);
```

Probajte rastezati granice appleta i vidjet ćete da će ispis uvijek biti centriran.

Moguće je u init metodi postaviti trajnu boju pozadine s:

```
setBackground(Color.blue);
```

Ako želite vidjeti što applet čini u određenom trenutku možete ubaciti pozive metode `System.out.println`. To će ispisati poruku u DOS prozoru na uobičajen način.

Stavite npr.:

```
System.out.println(”paint called”);
```

na početak `paint` metode i onda ćete u DOS prozoru vidjeti koliko često windows manager poziva paint metodu.

---

10. Zadaci

1. Unesite i testirajte primjer 5.

2. Napišite grafički program koji će ispisati vaše ime u centru ekrana unutar plavog pravokutnika . Applet nazovite ImeApplet.java, a Html datoteku ImeApplet.txt

3. Napišite grafički program koji će zatražiti unos radijusa i zatim nacrtati krug s tim radijusom. Applet nazovite KrugApplet.java, a Html datoteku KrugApplet.txt