Laboration 6

Avsikt

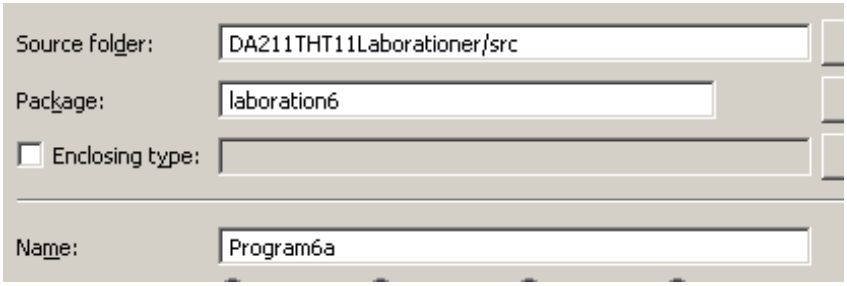
Avsikten är att du ska träna på att skriva små enkla program som använder färdiga klasser. Du ska också använda dig av ett speciellt program, debugger, vilket hjälper till att hitta felaktigheter i programmet du utvecklar.

I texten nedan följer ett antal program som ska skrivas. Skissa först på ett papper en möjlig lösning (t.ex. aktivitetsdiagram eller halvkod). När du är färdig med en uppgift ska du jämföra din lösning med den som är bifogad sist.

Grundläggande övningar

Uppgift 6a

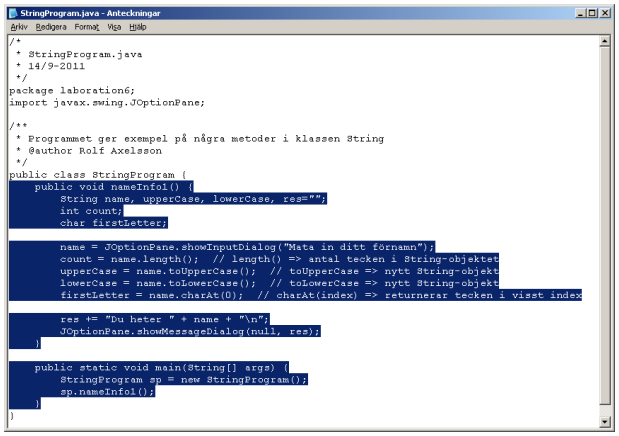
Nu ska du skapa klassen *Program6a* i paketet *laboration6*.



- Högerklicka *src* och välj New - Class
- Ange paket som klassen ska vara i: *laboration6*
- Ange klassens namn: *Program6a*

Kopiera innehåll från filen StringProgram.java (finns i laborations-filen) till klassen du just skapat.

Öppna StringProgram.java i Anteckningar, markera metoderna i klassen StringProgram och klistra in metoderna i klassen Program6a.




```
/*
 * StringProgram.java
 * 14/9-2011
 */
package laboration6;
import javax.swing.JOptionPane;

/**
 * Programmet ger exempel på några metoder i klassen String
 * @author Rolf Axelsson
 */
public class StringProgram {
    public void nameInfo1() {
        String name, upperCase, lowerCase, res="";
        int count;
        char firstLetter;

        name = JOptionPane.showInputDialog("Mata in ditt förnamn");
        count = name.length();  // length() => antal tecken i String-objektet
        upperCase = name.toUpperCase();  // toUpperCase => nytt String-objekt
        lowerCase = name.toLowerCase();  // toLowerCase => nytt String-objekt
        firstLetter = name.charAt(0);  // charAt(index) => returnerar tecken i visst index

        res += "Du heter " + name + "\n";
        JOptionPane.showMessageDialog(null, res);
    }

    public static void main(String[] args) {
        StringProgram sp = new StringProgram();
        sp.nameInfo1();
    }
}
```

```
package laboration6;

public class Program6a {
    public void nameInfo1() {
        String name, upperCase, lowerCase, res="";
        int count;
        char firstLetter;

        name = JOptionPane.showInputDialog("Mata in ditt förnamn");
        count = name.length();  // length() => antal tecken i String-objekt
        upperCase = name.toUpperCase();  // toUpperCase => nytt String-obje
        lowerCase = name.toLowerCase();  // toLowerCase => nytt String-obje
        firstLetter = name.charAt(0);  // charAt(index) => returnerar tecke

        res += "Du heter " + name + "\n";
        JOptionPane.showMessageDialog(null, res);
    }

    public static void main(String[] args) {
        StringProgram sp = new StringProgram();
        sp.nameInfo1();
    }
}
```

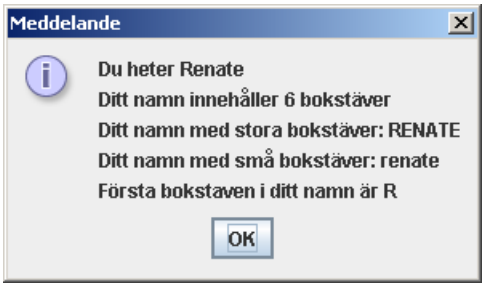
Vissa ändringar och tillägg måste du göra efter kopiering:

- Klassen JOptionPane måste importeras.
  `import javax.swing.JOptionPane;`
- I main-metoden ska ett objekt av typen Program6a skapas
  `Program6a sp = new Program6a();`

Nu bör rödmarkeringarna försvunnit och det är dags att köra programmet. Du får ett körresultat liknande dialogen till höger.

Men meningen är att dialogen ska innehålla mer information. Lägg till instruktioner vilka bygger upp strängen *res* så att dialogen visar information som fönstret till höger. Det behövs en eller flera rader liknanden:

```
res += ….;    // …. Ersätts med vettigt innehåll
```



Uppgift 6b

Skapa klassen ***Program6b*** och kopiera nedanstående innehåll till klassen. När du klistrat in innehållet så ska du formattera innehållet. Högerklicka *Program6b.java* och välj *Source – Format*.

Din uppgift är att komplettera metoden *date()* så att dagens datum skrivs ut i en meddelandedialog. Till din hjälp har du ***Calendar***-klassen och föreläsningsexemplet CalendarEx.java. Konstanter som är av intresse finner du i föreläsningsunderlaget.

Tänk på att månadsnumret som returneras är ett för litet. Du måste addera till 1 innan du gör utskriften.

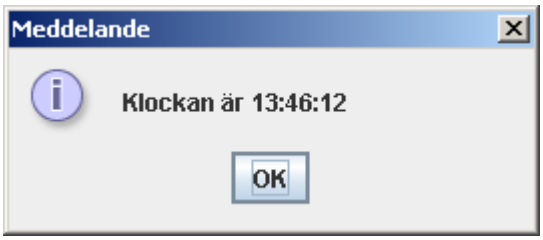
```
public class Program6b {
    public void date() {

    }

    public static void main(String[] args) {
        Program6b prog = new Program6b();
        prog.date();
    }
}
```

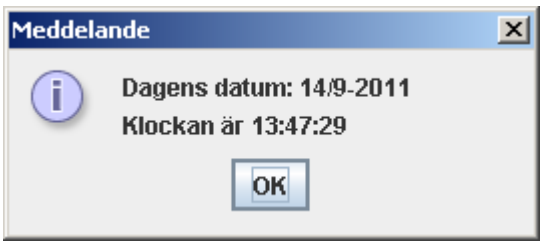
Uppgift 6c

Du ska komplettera klassen ***Program6b*** med metoden *time()* vilken ska skriva ut klockslaget i en meddelandedialog. Om du anropar metoden *time* ska du få ett körresultatet liknande dialogen till höger.

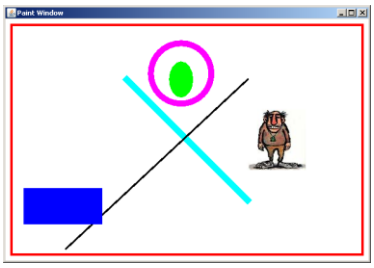


Uppgift 6d

Du ska komplettera klassen ***Program6b*** med metoden *dateAndTime()* vilken skriver ut dagens datum och aktuellt klockslag i en meddelandedialog. Vid anrop till metoden *dateAndTime* ska det bli ett körresultat liknande dialogen till höger.



Uppgift 6e



Hämta filerna *PaintWindow.java* och *PaintWindowDemo.java* från kurssidan och klistra in dem i katalogen laboration6. När du kör *PaintWindowDemo* (gör det!) så visar sig ett fönster som det till höger.

Tittar du i klassen *PaintWindowDemo*, metoden *demo*, så ser du anropen som ger fönstret sitt innehåll. De förklaras nu ett efter ett:

```
PaintWindow window = new PaintWindow();
```
Ett objekt av typen PaintWindow skapas. Referensvariabeln *window* håller reda på objektet.

```
ImageIcon image = new ImageIcon("M:/bilder/gubbe.jpg");
```
Ett ImageIcon-objekt skapas. Objektet innehåller angiven bild (sökvägen på hårddisken + filnamnet). Finns inte bilden på angivet ställe så innehåller objektet ej någon bild.
Bilden visas i fönstret genom anrop till metoden showImage (förklaras längre ner)

```
window.line(200, 100, 400, 300, Color.CYAN, 10);
```
En linje ritas i fönstret, från punkten (200,100) till punkten (400,300). Linjen har färgen Color.CYAN och är 10 pixlar bred.

```
window.line(100, 380, 400, 100, Color.BLACK, 3);
```
En linje ritas I fönstret, från punkten (100,380) till punkten (400,100). Linjen har färgen Color.BLACK och är 3 pixlar bred.

```
window.drawRect(10, 10, 580, 380, Color.RED, 4);
```
En rektangel ritas i fönstret. Rektangelns övre vänstra hörn är i punkten (10,10). Rektangeln är 580 pixlar bred och 380 pixlar hög. Rektangeln ör i färgen Color.RED och rektangelns sida är 4 pixlar bred.

```
window.fillRect(30, 280, 130, 60, Color.BLUE);
```
En fylld rektangel ritas i fönstret. Rektangelns övre vänstra hörn är i punkten (30,280). Rektangeln är 130 pixlar bred och 60 pixlar hög. Rektangeln har färgen Color.BLUE.

```
window.drawOval(240, 40, 100, 100, Color.MAGENTA, 10);
```
En ellips ritas i fönstret. Ellipsen är ”inskriven” i en angiven rektangel med övre vänstra hörnet i punkten 240,40, bredden 100 pixlar och höjden 100 pixlar. Färgen på ellipsen är Color.MAGENTA och ellipsens sida är 10 pixlar bred.

```
window.fillOval(270, 70, 40, 60, Color.GREEN);
```
En fylld ellips ritas i fönstret. Ellipsen är inskriven i en angiven rektangel med övre vänstra hörnet i punkten (270,70), bredden 40 pixlar och höjden 60 pixlar.

```
window.showImage(image, 400, 150);
```
Bilden som image refererar till visas i fönstret. Bildens övre vänstra hörn är i punkten (400,150).

Ändra i metoden demo så att en i ditt tycke snygg teckning skapas ☺.

Uppgift 6f

Skapa klassen ***Program6f***. Skriv metoden *images* vilken ska göra följande:

Skapa ett PaintWindow-objekt (som i Uppgift 6e)

Läsa in några olika bilder från hårddisken. Du kan själv hämta bilderna från nätet. De ska ha något av formaten jpg, gif eller png.

```
ImageIcon image1 = new Image("M:/bilder/Summer.jpg");
ImageIcon image2 = new Image("M:/bilder/Winter.jpg");
```

Gör ett snyggt collage av bilderna i PaintWindow-objektet.

Att finna fel i ett program

Hämta **Program6g.java** från kurssidan.Nu ska du testköra programmet med nedanstående inmatningar. *nbr1* kommer få ett slumpvärde i intervallet 1-10. I exemplet nedan är slumpvärdet 7.

Vid `"Ange ditt namn"` mata in `Kasper`
Vid `"Ange ett tal"` mata in `0`

Vad hände efter den sista inmatningen?

```
Kasper, så här blir det:
Addition:                   7+0=7
Subtraktion:                7-0=7
Multiplikation:             7*0=0
Exception in thread "main" Division:                    7/0=Infinity
java.lang.ArithmeticException: / by zero
        at laboration6.Program6g.calculate(Program6g.java:21)
        at laboration6.Program6g.main(Program6g.java:27)
```

För att se radnummer i Eclipse så välj

Window – Preferences
General – Editors – Text Editors

och markera Show line numbers

Ett felmeddelande skrivs ut av systemet (röd text). Dett innebär att något gått fel.

Om vi tittar på felmeddelandet så får vi viss information om vad som hänt:

- `java.lang.ArithmeticException: / by zero`
  division med 0.
- `at laboration6.Program6g.calculate(Program6g.java:21)`
  felet inträffade på rad 21.

Programmet klarade inte att dividera med 0 vid heltalsdivision (går bra med decimaltal). Så fort ett fel inträffar så avbryts programexekveringen.

Ofta är det svårt att förstå varför ett program avslutas av ett fel eller inte ger det körresultat som avsetts. Två tänkbara strategier för att finna felaktigheterna är:

1. Lägga in utskriftrader på kritiska platser i koden. Genom att skriva ut variablers värden i olika skeden i programmet kan ett och annat klarna.

```
double sum=0;
int nbr;
nbr = Input.readInt("Mata in ett heltal");
sum += nbr /2;
System.out.println(sum);  // Undersökning av värdet på summa efter varje addition
nbr = Input.readInt("Mata in ett heltal");
sum += nbr /2;
System.out.println(sum);  // Undersökning av värdet på summa efter varje addition
```

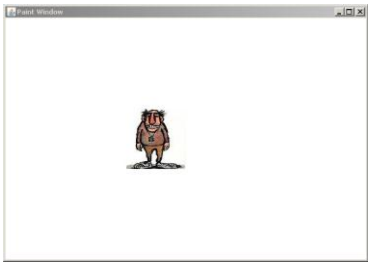
2. Använda en **debugger**. Om detta handlar nästa avsnitt.

Nu ska du arbeta med bilaga 1, **Använda debugger i Eclipse**

Om du använder en annan utvecklingsmiljö än ovanstående får du på egen hand undersöka möjligheterna att debugga program.

Fördjupande övningar

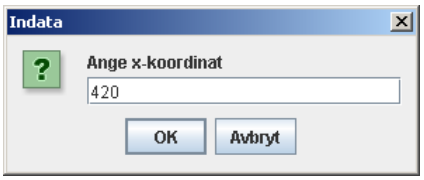
Uppgift 6h



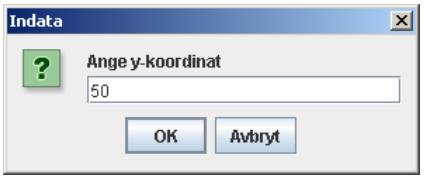
Om man visar en bild i *PaintWindow* så går det bra att flytta bilden genom ett nytt anrop till *showImage*, men med andra koordinater.

Skriv ett program vilket

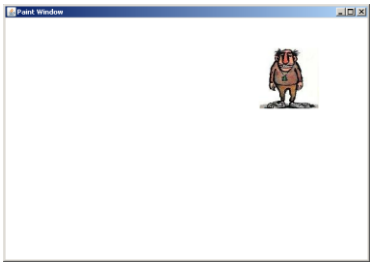
- visar en bild i ett PaintWindow-objekt



- visar en inmatningdialog i vilken användaren ska mata in en x-koordinat



- visar en inmatningdialog i vilken användaren ska mata in en y-koordinat



- anropar *showImage*-metoden med de nya koordinaterna

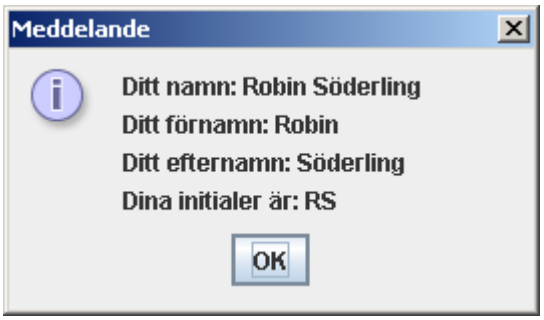
Uppgift 6i

Nu ska du skriva ett program vilket ska utföra följande:

Låta användaren mata in sitt namn (förnamn och efternamn) i en inmatningdialog

Tar reda på i vilken position blanktecknet i den inmatade strängen finns (luckan mellan förnamn och efternamn) – använd metoden `indexOf( char )`

```
int index = name.indexOf(' '); // Ex name="Rut Al" => index=3
```



Skriver ut personens förnamn – använd metoden `substring(startpos, slutpos-1)`

```
String str = "En varm sommar";
String nyStr = str.substring(3,7); //
nyStr="varm", tecknen 3-6
```

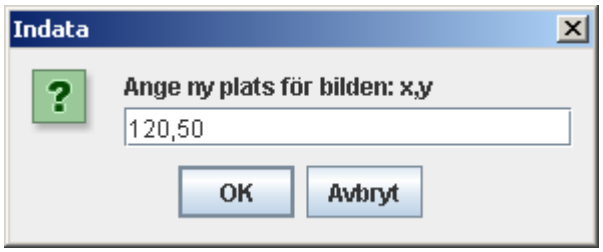
Skriver ut personens efternamn – använd metoden `substring( startpos )`

```
String str = "En varm sommar";
String nyStr = str.substring(8); //
nyStr="sommar", tecknen 8-slutet
```

Skriver ut personens initialer – använd `charAt( pos )` två gånger

Uppgift 6j

Skriv ett program liknande det i *Uppgift 6h*. Skillnaden är att användaren ska mata in x- och y-koordinat i ett och samma inmatningsfönster.



Dina kunskaper från *Uppgift 6i* kommer till användning i din lösning.

Program6k

Skriv ett program vilket visar en bild i en slumpmässig position i PaintWindow. Välj slumpvärde så att det hamnar i intervallet:

x-värde: 0 till (600-bildens bredd). Bildens bredd ges genom anrop till metoden *getIconWidth()* i ImageIcon-objektet.

```
ImageIcon image = new ImageIcon("M:/bilder/bok.jpg");
int bredd = image.getIconWidth();
```

y-värde: 0 till (400-bildens hörd). Bildens höjd ges genom anrop till metoden *getIconHeight()* i ImageIcon-objektet.

Testkör din lösning ett antal gånger. Vid varje körning ska hela bilden vara synlig i fönstret och dessutom hamna i olika positioner.

Extra övningar

Uppgift 6l

Ändra i din lösning i Uppgift 6j så att alltid hela bilden visas, oavsett vilka koordinater användaren matar in.

Det innebär att x-koordinaten aldrig ska vara mindre än 0.
Det innebär att x-koordinaten aldrig ska vara större än (600 – bildens bredd)

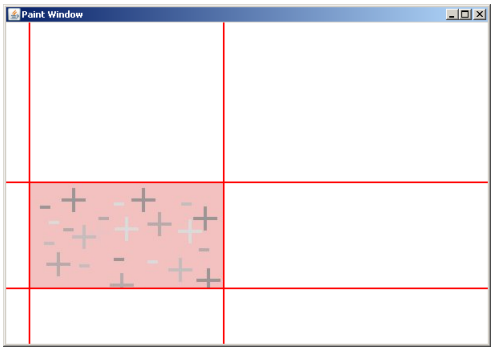
Det innebär att y-koordinaten aldrig ska vara mindre än 0.
Det innebär att y-koordinaten aldrig ska vara större än (400 – bildens höjd)

Exempel:

Ett inmatat värde ska vara i intervallet 2 – 5. Om användaren matar in ett värde mindre än två ska värdet ändras till 2. Om användaren matar in ett värde större än 5 ska värdet ändras till 5. Metoderna Math.min och Math.max kommer att användas.

```
int tal = Integer.parseInt(JOptionPane.showInputDialog("Mata in ett tall, 2-5"));
tal = Math.max(tal, 2); // tal tilldelas det största värdet av 2 och det värde tal har
tal = Math.min(tal, 5); // tal tilldelas det minsta värdet av 5 och det värde tal har
```

Du kan leka vidare och ”rama in” bilden med vertikala resp horisontella röda linjer. Linjerna kan ha bredden 2 pixlar.



Uppgift 6m

Om du av någon anledning vill ha reda på bildskärmens upplösning kan du använda klassen ***Toolkit***. Ett Toolkit-objekt får man genom anrop till klassmetoden *Toolkit.getDefaultToolkit()*.
Skärmens upplösning får du genom anrop till metoden *getScreenSize()*.

Värdet som returneras är av typen ***Dimension***. Ett objekt av typen Dimension innehåller två värden, *width* och *height*. Dessa värden kan man dirket åtkomst till.

Båda klasserna finns i paketet *java.util*.

Exempel

```
int width, height;
Toolkit toolkit = Toolkit.getDefaultToolkit();
Dimension d = toolkit.getScreenSize();
width = d.width;
height = d.height;
```

Skriv ett program som meddelar skärmens upplösning i en meddelandedialog.

.

Lösningar

Program6a

```
package laboration6;
import javax.swing.JOptionPane;

public class Program6a {
    public void nameInfo1() {
        String name, upperCase, lowerCase, res="";
        int count;
        char firstLetter;

        name = JOptionPane.showInputDialog("Mata in ditt förnamn");
        count = name.length();  // length() => antal tecken i String-objektet
        upperCase = name.toUpperCase();  // toUpperCase => nytt String-objekt
        lowerCase = name.toLowerCase();  // toLowerCase => nytt String-objekt
        firstLetter = name.charAt(0);  // charAt(index) => tecken i visst index

        res += "Du heter " + name + "\n";
        res += "Ditt namn innehåller " + count + " bokstäver\n";
        res += "Ditt namn med stora bokstäver: " + upperCase + "\n";
        res += "Ditt namn med små bokstäver: " + lowerCase + "\n";
        res += "Första bokstaven i ditt namn: " + firstLetter;
        JOptionPane.showMessageDialog(null, res);
    }

    public static void main(String[] args) {
        Program6a sp = new Program6a();
        sp.nameInfo1();
    }
}
```

Program 6b – 6d

```
package laboration6;
import java.util.Calendar;
import javax.swing.JOptionPane;

public class Program6b {
    public void date() {
        Calendar cal = Calendar.getInstance();
        String message = "Dagens datum: " + cal.get(Calendar.DAY_OF_MONTH) +
                "/" + (cal.get(Calendar.MONTH) + 1) + "-" + cal.get(Calendar.YEAR);
        JOptionPane.showMessageDialog(null, message);
    }
    public void time() {
        Calendar cal = Calendar.getInstance();
        String message = "Klockan är " + cal.get(Calendar.HOUR_OF_DAY) + ":" +
                cal.get(Calendar.MINUTE) + ":" + cal.get(Calendar.SECOND);
        JOptionPane.showMessageDialog(null, message);
    }
    public void dateAndTime() {
        Calendar cal = Calendar.getInstance();
        String message = "Dagens datum: " + cal.get(Calendar.DAY_OF_MONTH) +
                "/" + (cal.get(Calendar.MONTH) + 1) + "-" + cal.get(Calendar.YEAR);
        message += "\n";
        message += "Klockan är " + cal.get(Calendar.HOUR_OF_DAY) + ":" +
                cal.get(Calendar.MINUTE) + ":" + cal.get(Calendar.SECOND);
        JOptionPane.showMessageDialog(null, message);

    }
    public static void main(String[] args) {
        Program6b prog = new Program6b();
        prog.date();
        // prog.time();
        // prog.dateAndTime();
    }
}
```

Program 6h

```
package laboration6;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class Program6h {
    public void movingMan() {
        PaintWindow window = new PaintWindow();
        int x,y;
        String str;

        ImageIcon image = new ImageIcon("M:/bilder/gubbe.jpg");
        window.showImage(image, 150, 150);
        str = JOptionPane.showInputDialog("Ange x-koordinat");
        x = Integer.parseInt(str);
        str = JOptionPane.showInputDialog("Ange y-koordinat");
        y = Integer.parseInt(str);
        window.showImage(image, x, y);
    }

    public static void main(String[] args) {
       Program6h prog = new Program6h();
        prog.movingMan();
    }
}
```

Program 6i

```
package laboration6;
import javax.swing.JOptionPane;

public class Program6i {
    public void action() {
        String name, firstName, lastName, initials, message;
        int whiteSpace;
        name = JOptionPane.showInputDialog("mata in förnamn och efternamn");
        whiteSpace = name.indexOf(' ');
        firstName = name.substring(0,whiteSpace);
        lastName = name.substring(whiteSpace+1);
        initials = "" + name.charAt(0) + name.charAt(whiteSpace+1);
        message = "Ditt namn: " + name + "\n" +
                  "Ditt förnamn: " + firstName + "\n" +
                  "Ditt efternamn: " + lastName + "\n" +
                  "Dina initialer är: " + initials;
        JOptionPane.showMessageDialog(null, message);
    }
    public static void main(String[] args) {
        Program6i prog = new Program6i();
        prog.action();
    }
}
```

Program 6j

```
package laboration6;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class Program6j {
    public void movingMan() {
        PaintWindow window = new PaintWindow();
        int x,y,comma;
        String str,xPart,yPart;

        ImageIcon image = new ImageIcon("M:/bilder/gubbe.jpg");
        window.showImage(image, 150, 150);
        str = JOptionPane.showInputDialog("Ange ny plats för bilden: x,y");
        comma = str.indexOf(',');
        xPart = str.substring(0,comma);
        x = Integer.parseInt(xPart);
        yPart = str.substring(comma+1);
        y = Integer.parseInt(yPart);
        window.showImage(image, x, y);
    }

    public static void main(String[] args) {
       Program6j prog = new Program6j();
        prog.movingMan();
    }
}
```

Program 6k

```
package laboration6;
import javax.swing.ImageIcon;
import java.util.Random;

public class Program6k {
    public void randomPosition() {
        PaintWindow window = new PaintWindow();
        Random rand = new Random();
        int x,y;

        ImageIcon image = new ImageIcon("M:/bilder/gubbe.jpg");
        window.showImage(image, 150, 150);
        x = rand.nextInt(600-image.getIconWidth());
        y = rand.nextInt(400-image.getIconHeight());
        window.showImage(image, x, y);
    }

    public static void main(String[] args) {
       Program6k prog = new Program6k();
        prog.randomPosition();
    }
}
```

Program 6l

```
package laboration6;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class Program6l {
    public void movingMan() {
        PaintWindow window = new PaintWindow();
        int x,y,comma;
        String str,xPart,yPart;

        ImageIcon image = new ImageIcon("M:/bilder/gubbe.jpg");
        window.showImage(image, 150, 150);
        str = JOptionPane.showInputDialog("Ange ny plats för bilden: x,y");
        comma = str.indexOf(',');
        xPart = str.substring(0,comma);
        x = Math.max( Integer.parseInt(xPart), 0);
        x = Math.min( x, 600-image.getIconWidth());
        yPart = str.substring(comma+1);
        y = Math.max( Integer.parseInt(yPart), 0);
        y = Math.min( y, 400-image.getIconHeight());
        window.showImage(image, x, y);
    }

    public static void main(String[] args) {
       Program6l prog = new Program6l();
        prog.movingMan();
    }
}
```

Program 6m

```
package laboration6;
import java.awt.*;
import javax.swing.JOptionPane;

public class Program6m {
    public static void main(String[] args) {
        int width, height;
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Dimension d = toolkit.getScreenSize();
        width = d.width;
        height = d.height;
        JOptionPane.showMessageDialog(null, "Skärmupplösning: " + width + "x" +
height);
    }
}
```

Bilaga 1

Använda debugger i Eclipse

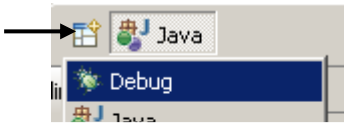
I programmeringsmiljöer finns det oftast någon form av debugger. Den är till för att man ska hitta felaktigheter i ett program. Med debuggern i Eclipse kan man bl.a.:

- Följa exekveringen sats för sats. Man kan alltså se i vilken ordning satser utförs och om de över huvud taget utförs. Man använder knapparna *Step Over*, *Step Into* och *Step Return*.
- Se vilket värde variabler har i olika skeden i programkörningen. Man använder *Variable*-fönstret för att se variabelvärden.
- Stanna programmet vid speciella satser och där studera bl.a. variablers värde. Man använder breakpoints tillsammans med *Variable*-fönstret.

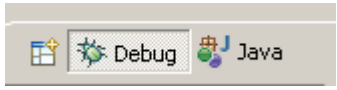
Hämta filen **Guess.java** från kurssidan och placera den i din projektkatalog. Kör programmet ett par gånger så du ser hur det fungerar.

Debug File

När du använder debuggern ska du ge Eclipse ett speciellt utseende, perspective. Klicka på *Open perspective*-knappen uppe till höger i Eclipse. Välj *Debug* i popup-menyn.



Nu ändras utseendet på Eclipse och du ser bl.a. ett fönster som heter **Debug**. Du kan ändra perspektiv genom att klicka på knapparna upp till höger (Java /Debug). Klicka först på Java, och sedan på Debug.



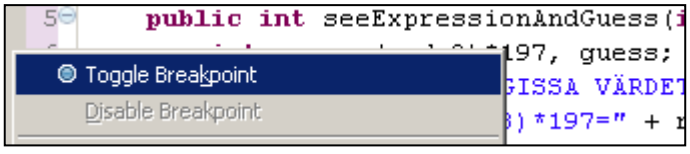
Starta programmet genom att *högerklicka* i editorn och välj **Debug As – 1 Java Application** i popup-menyn.

Programmet kommer att exekvera helt vanligt.

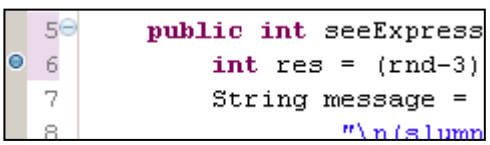
Breakpoints

För att få programkörningen att pausa mitt i en exekvering kan du sätta en eller flera **breakpoints**. Om du startar ett program med **Debug As – 1 Java Application** så sker programkörningen normalt tills en breakpoint nås. Då avbryts programkörningen och man kan studera t.ex. variablers värde just i det skedet.

Det är enkelt att sätta breakpoints i ett program / ta bort breakpoints ur ett program. Man *högerklickar* i vänstra marginalen och väljer *Toggle Breakpoint*. Klicket ska ske vid den rad där man vill att programkörningen ska pausas.



På rad 6 är det ingen breakpoint. Men om du *högerklickar* + *Toggle Breakpoint* med musmarkören ungefär ovanpå 6:an så sätter man en breakpoint på rad 6.



Sedan tar man bort den med ytterligare ett klick.

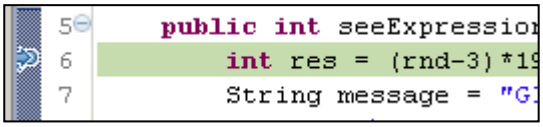
När programmet pausar vid en breakpoint har man bl.a. 5 olika valmöjligheter. Dessa är på en knappanel ovanför debug-fönstret (och i run-menyn).

1 2 3 4 5

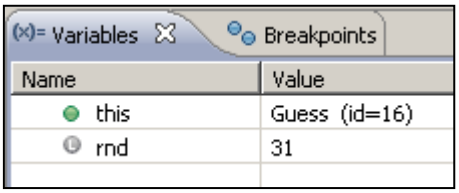
1. Exekvera till nästa breakpoint.
2. Avbryt programmet
3. Exekvera markerad sats. (**Step Into** – gå in i metod vid metodanrop på raden)
4. Exekvera markerad sats. (**Step Over –** hoppar över metoder vid metodanrop på raden)
5. Exekvera metod. (**Step Return** – exekverar färdigt den metod där programmet pausat)

Exekvera

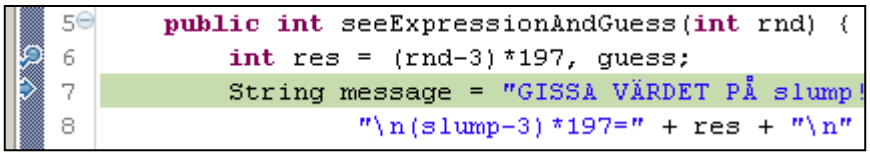
Sätt en breakpoint på rad 6 (i Guess.java), högerklicka ovanpå editorn och välj **Debug As – 1 Java Application**. Nu startar programmet för att pausas *innan* rad 6 utförs.



Om du studerar fönstret **Variables** så kan du se att variabeln *rnd* har fått ett värde. Variabler visar sig i debuggern först då de har ett värde.



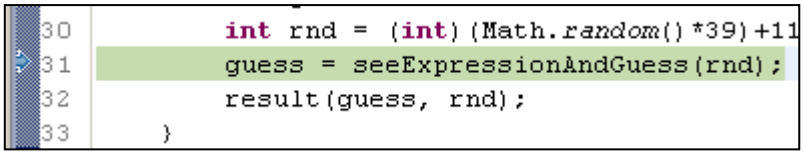
Exekvera rad 6 genom att klicka på **Step Over** (nr 4). Nu flyttar sig den gröna markeringen till rad 7. Samtidigt ser du att variabeln *res* dök upp i *Variables*-fönstret. Detta beror på att *res* tilldelades ett värde för första gången på rad 6.



Klicka på **Step Over**. Nu fortsätter exekveringen till rad 8. Fortsätt att klicka på **Step Over** tills rad 11 är grön. Nu ser du i *Variables*-fönstret att variabeln *message* fått ett värde.

Rad 11 innebär att en dialog ska visas och att programmet väntar tills användaren klickat på OK. Om du inte ser inmatningsfönstret så döljs det av andra fönster. Håll **Alt**-knappen nere och klicka på **tab**-tangenten. Fortsätt att klicka med tab-tangenten tills rektangeln är ovanpå java-symbolen (kopp med rykande kaffe, syns inte koppen så var du för snabb och välj Eclipse och upprepa sökandet). Nu kommer inmatningsdialogen att visa sig.

Skriv in ett tal och klicka på **OK**. Nu blir rad 12 grönfärgad och står i tur att exekveras. Om du klickar en gång till på **Step Over** så tar metoden *seeExpressionAndGuess* slut och den rad där anropet till metoden skedde blir grönmarkerad.



Klickar du en gång på **Step Over** så flyttar sig exekveringen till rad 32. På rad 32 är det ett nytt metodanrop. Om du vill följa exekveringen steg för steg i metoden ska du nu välj **Step Into**. Vill du att hela metoden ska exekveras i ett steg så ska du välja **Step Over**.

Om du är inuti en metod och vill att metoden ska exekvera färdigt ska du välja **Step Return**. **Step Return** kan du även använda för att komma tillbaka till editor-fönstret om du klickat på **Step Into** när en metod i en standard-klass anropas.

När sista raden exekverats / eller du vill att programmet ska exekvera klart / avbrytas så ska du klicka på någon av knapparna 1 eller 2 ovan.