Week 6
Arrays
Applets
Function Decomposition

Why do we need arrays ?

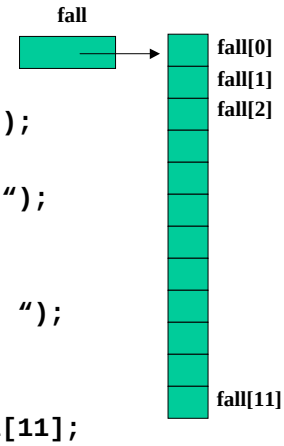
Given the rainfall for 12 months of 2000 compute the average.

```
double fall1, fall2, fall3, …. fall12;
System.out.prinltn("R.Fall month 1 : ");
fall1 = console.readInt();
System.out.prinltn("R.Fall month 2 : ");
fall2 = console.readInt();
...
System.out.prinltn("R.Fall month 12 : ");
fall12 = console.readInt();
double total = fall1 + fall2 + … fall12;
double average = total/12;
```

Too many variables. Prone to errors. Not practical !

Arrays help us avoid declaring variables

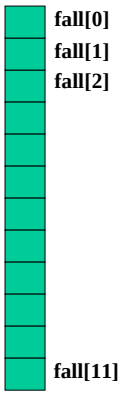
```
double[] fall;
fall = new double[12];
System.out.prinltn(“R.Fall month 1 : “);
fall[0] = console.readDouble();
System.out.prinltn(“R.Fall  month 2 : “);
fall[1] = console.readDouble();
...
System.out.prinltn(“R.Fall  month 12 : “);
fall[11] = console.readDouble();

double total = fall[0]+fall[1]... fall[11];
double average = total/12;
```



Does it make life easy ? Hardly! Many repetitive statements !

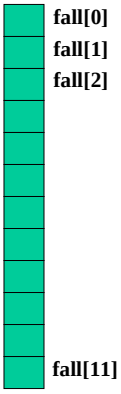
Using a loop and a control variable ...

```
double[] fall;
fall = new double[12];
for (int i =__ ; i < __; i++ ) {
  System.out.prinltn(“R.Fall month”+i+”: “);
       = console.readDouble();
}
double total = __;
for (int i =__ ; i < __; i++ )
  total += _________;
average = total/12;
```



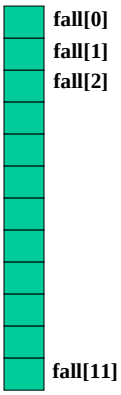
Quiz

```
// sums even months Feb,Apr,June ...
for (int i = __ ; i < __; ____)
  total += fall[i];


for (int i =0; i < 7; i +=2 )
  total += fall[i];
// sums fall for months _______________
```



Reading and displaying in Reverse order

```
import java.io.*;
public class RainFall {
    public static void main(String[] args)  throws IOException  {
        BufferedReader stdin = new BufferedReader
                    (new InputStreamReader (System.in));
        String inString;
        int i;
        double[] fall = new double[12];
        for ( i = 0 ; i < 12 ; i++)  {
              System.out.print("Enter rainfall for month " + (i+1) + ": ");
              inString = stdin.readLine();
              fall[i] = Double.parseDouble(inString);
        }

        System.out.println("Rain Fall in Reverse Order ");
        for ( i = 11 ; i >= 0 ; i--)  {
             System.out.println("month "+(i+1)+"=" + fall[i]);
        }

    }
}
```



Coins program without repetition

```
public class Change {
  public static void main(String[] args) {
      ConsoleReader console = new ConsoleReader(System.in);
      int[] denoms = {200,100,50,20,10,5,1};
      String[] descriptions  = {"$2","$1","50C","20C","10C","5C","2C","1C" };

       System.out.print("Enter amount : ");
       int amount = console.readInt();

       int i=0, count=0, number;
       while  ( amount > 0 ) {
            number = amount/denoms[i];
            System.out.println(descriptions[i]  +  " coins =  " +  number);
            amount %= denoms[i];
            count += number;
            i++;
       }
       System.out.println("Number of coins = " + count);
   }
}
```



```
class Account {
  public Account(String accountID,
       String accountName, double amount)   {…}
  public void deposit(double amount) {… }
  public boolean withdraw(double amount) {…}
  public double getBalance() {…}
  public boolean transfer(Account a, double amt){…}
  …
}
class TestAccount{
   Account mum = new Account("Mercy Brown",1000);
   Account dad = new Account("David Brown",2000);
   …
```

What if there are many such objects ? Use an Array !!!

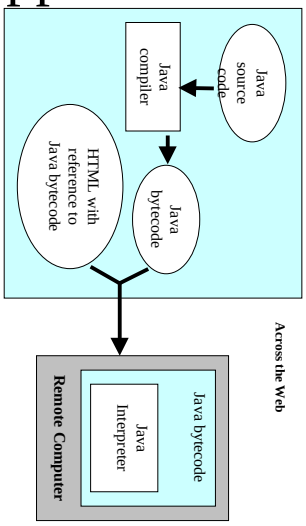
```
import java.io.*;
public class TestAccounts {
  public static void main(String args[]) {
     BufferedReader stdin = new BufferedReader(
             new InputStreamReader(System.in));
     String name, accountID;
     double balance;
     Account[] accounts = new Account[3];
     for (int i=0;i<3; i++) {
       System.out.println("ID Account : ” +(i+1));
       accountID = stdin.readLine();
       System.out.println("Name Account : ”    +(i+1));
       name = stdin.readLine();
       System.out.println("Init.Bal. Account : ” +(i+1));
       balance = Double.parseDouble(stdin.readLine());
       accounts[i] = new Account(accountID,name,balance);
     }
```



```
        // debit monthly charges
        for (int i=0;i<3; i++) {
               accounts[i].withdraw(5.00);
        }

        // print all account balances
        for (int i=0;i<3; i++) {
           System.out.println("Balance for Account " + (i+1)
                     + " is " + accounts[i].getBalance());
        }
     }
  }
```

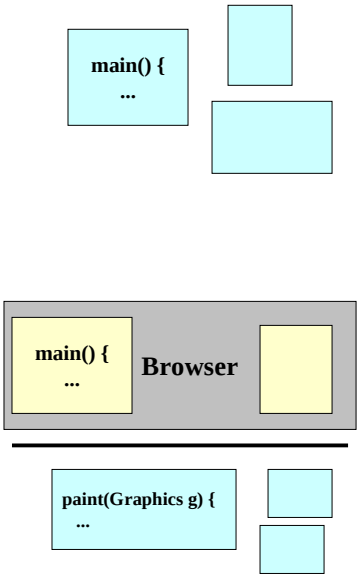
Introduction to applets

- **Two kinds of Java programs applets and applications.**
- **Applet (bytecode) is intended to be embedded into an HTML document, transported across the network and executed using a Web browser.**
- **So far, we have only looked at applications, the stand-alone programs that can be executed using the Java interpreter.**
- **Java applets can also be viewed locally using a Web browser or tool in Java's Software Development Kit, called appletviewer.**



Differences Applets & Applications

- **Unlike an application, applet is an incomplete program that is executed as part of another application, the browser.**
- **does not have a main() method, starting point for applications.**
- **The paint() method in the Applet derived class is automatically called whenever there is a need.**



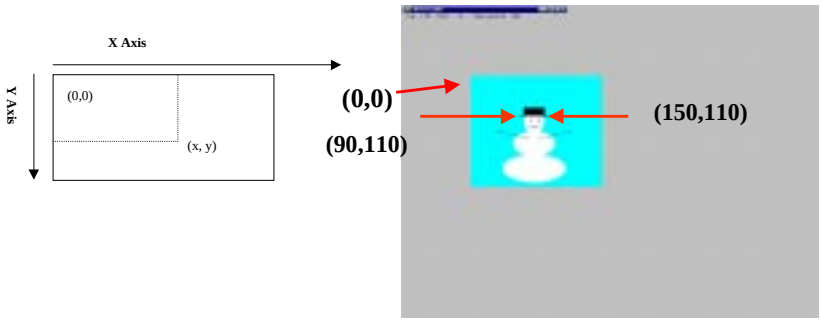
Creating a simple Applet

- Derive a class from Applet (1)
- Override the base class method paint(Graphics g) (2)
- draw on graphics object g (canvas) using g.drawRect(), g.drawLine(), ...

(2) (1)

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

public class Picture extends Applet {
  public void paint(Graphics g)   {        …
      g.setColor(Color.black);
      g.drawLine(90,110,150,110);// hat
      g.fillRect(100,90,40,20);  // hat
      g.fillOval(111,122,4,4);   // eye
      …
   }
}
```

Java Coordinate System




```
g.drawLine(90,110,150,110);// hat
g.fillRect(100,90,40,20);  // hat
g.fillOval(111,122,4,4);   // eye
```

Commonly used methods of Graphics class

void setColor(Color color)

drawString(String str, int x, int y)

void drawLine(int x1, int y1, int x2, int y2)

void drawOval(int x, int y, int width, int height)

void drawArc(int x, int y, int width, int height, int startAngle, int endAngle)

Adding color to life

- **To set color use void setColor(Color color)**
- **The predefined Color objects are objects are Color.black, Color.blue, Color.cyan, …**
- **Java uses the RGB color model, wherby you specify the color by the amounts (0.0F to 1.0F) of primary colors – red, green and blue.**
- **You may create your own color by mixing them in the required proportions as in**

```
Color magenta = new Color(1.0F, 0.0F, 1.0F);
```

Executing Applets using the Web

- To transmit applet over the Web we need a (HTML) program.
- It contains tags specifying formatting instructions and identifying special types such as applet.
- HTML file below uses an applet tag to indicates that the bytecode stored in `Picture.class` should be transported over the network and executed on the remote machine viewing it.
- Notice the tag also indicates the height and width of the applet.
- To view the applet you can use the command `appletviewer Picture.html` or use a browser.

File Picture.html

```
<applet code="Picture.class" width=350, height=350>
</applet>
```

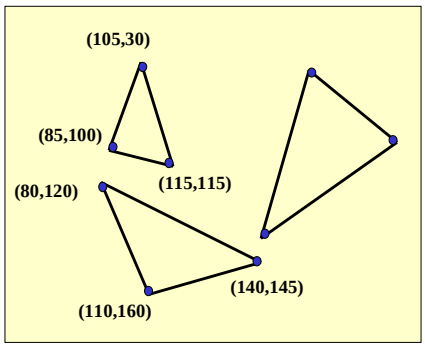

Method Decomposition

We are required to create a graphical application with 100's of triangles at user specified locations.

For each triangle we need to call drawLine() __ times.

Is there a better way ?

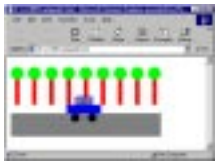
We can create a method say drawTriangle(), which takes the (x,y) coordinates of all 3 corners of that triangle. This method will then call drawLine() 3 times



Decomposing drawTriangle()

```
void drawTriangle(Graphics g, int x1,int y1,
                  int x2, int y2, int x3, int y3) {
      g.drawLine( ____________________________ );
      g.drawLine( ____________________________ );
      g.drawLine( ____________________________ );
}
//  paint method
drawTriangle(g,105,30,85,100,115,115);
drawTriangle(g,80,120,110,160,140,145);
```


An animation in 20 steps



```
//can write 20 methods - error prone - inefficient
drawCarPosition1(g); // draws at 1st position
drawCarPosition2(g); // draws at 2nd position
…
drawCarPosition20(g); // draws at 20th position
```

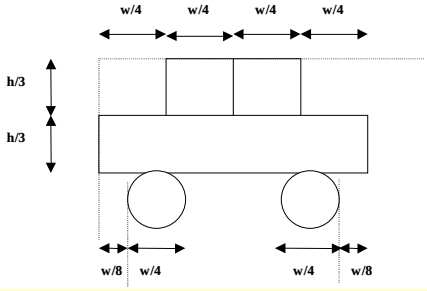
```
for (int i=0; i<20; i++){
   . . .
   drawCar(g, xpos, ypos, width, height, color);
}
```

Calls sub-methods

```
drawBody(g,...);
drawWindow(g,...);
drawWindow(g,...);
drawTyre(g,...);
drawTyre(g,...);
```

Decomposing drawCar()

h = height of car
w = width of car




```
public void drawCar(Graphics g, int x,int y, int width,
                                          int height, Color col)  {
        drawBody(g,x,y+height/3,width,height/3,col);
        drawWindow(g,x+width/4,y,width/4,height/3,col);
        drawWindow(g,x+width/2,y,width/4,height/3,col);
        drawTyre(g,x+width/8, y+2*height/3,width/4);
        drawTyre(g,x+5*width/8, y+2*height/3,width/4);
 }
```