CS 535 Object-Oriented Programming & Design
Fall Semester, 2000
Doc 9 An Example
Contents

Reference

Java API documentation

An Example
Point

```
public class Point{
   private float x;
   private float y;

   public Point( float x, float y) {
      this.x = x;
      this.y = y;
   }

   public Point add( Point aPoint){
      return new Point( x + aPoint.x, y + aPoint.y);
   }

   public double distance( Point aPoint) {
      float deltaX = x - aPoint.x;
      float deltaY = y - aPoint.y;
      return Math.sqrt( deltaX* deltaX + deltaY* deltaY);
   }

   public boolean greaterThan(Point  aPoint) {
      return (x > aPoint.x ) && (y  > aPoint.y);
   }

   public String toString() {
      return "(" + x + ", " + y + ")";
   }

   public boolean equals( Point p) {
      return x == p.x && y == p.y;
   }
}
```

Using Point

```
public class Sample {
   public static void main(String args[])  {
      Point a = new Point( 1, 0);
      Point b = new Point( 3, -1);
      Point c = a.add( b);
      System.out.println( a + " + " + b + " = " + c);
      System.out.println( "The distance between " + a + " and " +
          b + " is " + a.distance( b ));
   }
}
```

Output

```
(1.0, 0.0) + (3.0, -1.0) = (4.0, -1.0)
The distance between (1.0, 0.0) and (3.0, -1.0) is
2.23606797749979
```

toString() Standard

When an object is added to a string

- Object is converted to a string by calling its toString() method
- The two strings are then concatenated

println converts objects to strings via toString()

```
Point a = new Point( 1, 0);
System.out.println( a );
```

PointOperations

```
public class PointOperations {
   public String toString(float x, float y) {
      return "(" + x + ", " + y + ")";
   }

   public boolean greaterThan(float x, float y, float px, float py) {
      return (x > px ) && (y  > py);
   }

   public boolean equals(float x ,float y, float px , float py) {
      return x == px && y == py;
   }

   public double distance(float x, float y, float px ,float py) {
      float deltaX = x - px;
      float deltaY = y - py;
      return Math.sqrt( deltaX* deltaX + deltaY* deltaY);
   }

   public float[] add( float x, float y, float px, float py) {
      float[] answer = new float[2];
      answer[0] = x + px;
      answer[1] = y + px;
      return answer;
   }
}
```

Using PointOperations

```
public class TrivialApplication {
   public static void main(String args[])  {
      float x1 = 1;
      float y1 = 0;
      float x2 = 3;
      float y2 = -1;

      PointOperations a = new PointOperations();
      float[] answer = a.add( x1, y1, x2, y2);
      System.out.println( a.toString( x1 , y1) + " + " +
         a.toString( x1, y2) + " = " + a.toString( answer[0] , answer[1]));
      System.out.println( "The distance between " + a.toString( x1 , y1) +
          " and " + a.toString( x1, y2) + " is " +
          a.distance( x1, y1, x2, y2 ));
      }
}
```

Output

```
(1.0, 0.0) + (1.0, -1.0) = (4.0, 3.0)
The distance between (1.0, 0.0) and (1.0, -1.0) is
2.23606797749979
```

Some Problems with PointOperations

Harder to use
Error prone
Less flexible

A Drawing Program

Build a drawing program that knows about Points & Rectangles

```
import java.awt.Graphics;

public interface Drawable {
   public void drawOn(Graphics display);
}

import java.awt.Graphics;

public class Point implements Drawable {

   public void drawOn( Graphics display) {
      display.fillOval(  Math.round(x),  Math.round(y),  3,  3  );
   }

   //rest of class as before
}
```

The Program

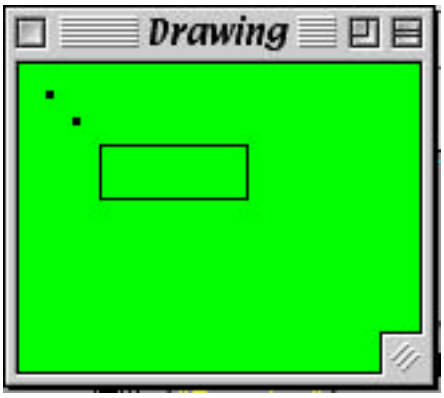
```
import  java.awt.*;
import java.util.*;

class  Drawing  extends  Frame {
   Vector pictureElements = new Vector();
   public void show( int  WidthInPixels, int heightInPixels ) {
      setTitle( "Drawing" );
      setSize( WidthInPixels, heightInPixels );
      setLocation( 40, 40);
      setBackground( Color.green);
      show();
   }

   public void add(Drawable element ){
      pictureElements.addElement( element );
   }

   public  void  paint(  Graphics  display  )  {
      Enumeration elements = pictureElements.elements();
      while (elements.hasMoreElements() ){
         Drawable x = (Drawable) elements.nextElement();
         x.drawOn( display);
      }
   }
   public  static  void  main( String  args[] ){
      Drawing example = new  Drawing(  );
      example.add( new Point( 10, 10));
      example.add( new Point( 20 , 20));
      example.add( new Rectangle( new Point( 30, 30), 20, 55));
      example.show( 150, 100);
   }
}
```

Program Output



In graphics the upper left corner is (0, 0) with positive y going down the screen

How could we write the program with PointOperations instead of Point?

Circle

```
import java.awt.Graphics;

public class Circle implements Drawable{
   Point center;
   float radius;

   public Circle( Point center, float radius){
      this.center = center;
      this.radius = radius;
   }

   public boolean includes(Point p ){
      return center.distance( p ) < radius;
   }

   public boolean equals( Circle aCircle ) {
      return center.equals(aCircle.center) && radius == aCircle.radius;
   }

   public void drawOn(Graphics display){
      display.drawOval( Math.round(center.x()), Math.round(center.y()),
         Math.round(radius), Math.round(radius));
      }

   public String toString(){
      return "Circle( center" + center + " raduis " + radius + " )";
      }
   }
```

Note requires accessor methods in point class.

Testing using text based TestRunner

```
import junit.framework.TestCase;
import junit.textui.TestRunner;

public class TestCircle extends TestCase {
   public static void main(String[] arg) {
      TestRunner.run( TestCircle.class);
   }
   public TestCircle(String name) {
      super( name);
   }

   public void testIncludes() {
      // {x , y}
      float[][] testData = { { 5, 5}, {6 , 6} , {7, 5} , {8, 8}};
      boolean[] answers = { true, true, false, false};
      Circle tester = new Circle( new Point( 5, 5), 2);
      for (int k = 0; k < testData.length; k++)
         {
         float x = testData[k][0];
         float y = testData[k][1];
         Point test = new Point( x, y);
         assert( "Data point " + k , answers[k] == tester.includes( test));
      }
   }
```

Testing Continued

```
public void testEquals()
  {
  // {centerX, centerY, radius}
  float[][] testData = { { 5.5f, 4.3f, 2},
    { 5.5f, 4.3f, 2.001f} ,
    { 5.5f, 5, 2} ,
    { 2, 1, 8}};
  boolean[] answers = { true, false, false, false};
  Circle tester = new Circle( new Point( 5.5f, 4.3f), 2);
  assert( "self " , tester.equals( tester));
  for (int k = 0; k < testData.length; k++)
    {
    Point center = new Point( testData[k][0], testData[k][1]);
    float radius = testData[k][2];
    Circle test = new Circle( center, radius);
    assert( "Data point " + k , answers[k] == tester.equals( test));
    }
  }
```

Running the Tests

```
java TestCircle
```

Output

```
..
Time: 0.485

OK (2 tests)
```

Some Java Info

Names

Use full words for names

origin
radius

Classes start with capital letter

Circle, Point

Methods, fields, & variables start with lowercase letter

Compiler Option

java -Djava.compiler=NONE className

Program runs slower, but provides better error messages

Use
setenv JAVA_COMPILER NONE
to set this option in your environment

Example

```
public class CompileExample {

   public static void main(String args[])
      {
      int a = 1;
      int b = 0;
      float c = a/b;
      }
   }
```

```
rohan 14-> javac CompileExample.java
rohan 15-> java CompileExample
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at CompileExample.main(Compiled Code)


rohan 16-> java -Djava.compiler=NONE  CompileExample
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at CompileExample.main(CompileExample.java:7)
```

printStackTrace

Prints the stack trace of an exception

```
public class StackTraceExample {
   public static void main(String args[])
      {
      try
         {
         Drawing.main( args);
         TestCircle.main(args);
         int a = 1;
         int b = 0;
         float c = a/b;
         }
      catch (Exception mathError)
         {
         mathError.printStackTrace( System.err);
         }
      }
   }
```