Midterm Exam

Intermediate Computing
CS 2530
Fall 2012

Instructions

- The exam consists of ten (10) questions and five (5) pages of code. Be sure that you have all of these items and that they are all legible.

- Read all questions and their instructions thoroughly before you begin. It is always worth your time to plan ahead!

- Write your answers in the spaces provided on the exam.

- Whenever you are asked to answer or describe "briefly", a two- (or maybe three-) sentence response should be enough to express the solution.

- Whenever you are asked to modify an existing piece of code, you only need to include the new and changes parts in your answer. Be sure to identify which part is which.

- All questions are of the same value. Use this information as you budget your time across the exam period.

- Points will be awarded based on your explicit answers. Partial credit will be given where possible, so show all of your work.

- The exam lasts seventy-five (75) minutes. It is due at 1:45 PM.

Name:

1. In object-oriented programming, we speak of **classes**, **instances**, **methods**, and **messages**. Briefly define and give an example of each.

2. Define the **instance variables** and the **constructor** for the `PriceRange` class, which implements the `Range` interface (see Page 1 of the code).

```
public class PriceRange implements Range {
```

3. Briefly answer these questions about the **Principle of Continuity**.

   - What is the Principle of Continuity?
   - Describe an example of how its use made one of our programs better, or an example of how not using it made one of our programs worse.

4. Write three JUnit tests for the `contains()` behavior of a `Range` object (see Page 1 of the code). Your assertions should examine distinct cases. You may place them in a single test method.

```
public class PriceRangeTest extends TestCase {
```

5. Briefly answer these questions about the `static` keyword in Java.

   - What is the difference between a variable in a class that is declared `static` and one that is not?
   - Why does the `main()` method have to be declared `static`?

6. Write the `overlaps()` and `size()` methods for the `PriceRange` class, based on the code you wrote for Problem 2.

```
public class PriceRange implements Range { ...
```

7. Briefly answer these questions about interfaces.

   - What is an interface?
   - How is extending a class different from implementing an interface?

8. Write a `GrowingDisk` class that extends the `Disk` class. A `GrowingDisk` behaves in all respects like any other `Disk`, except that every 100th time it `paint()`s itself, it grows one pixel bigger in every direction.

9. Two of the tools we used to document designs this semester are **object diagrams** and **interaction diagrams**. Briefly answer these questions about them.

   - Describe each kind of diagram.
   - What role does each play in describing the behavior of a program?

10. Modify `BallWorldFrame` so that, after going through its 20000-iteration cycle, it starts back at its beginning, creating a new `Ball` and going through another 20000 iterations.

```
/*
   A Range objects represents an integer range, such as
   1-10 or 50701-50799.  The lower and upper bounds of
   a Range are given at the time the object is created.
*/

public interface Range
{
  public boolean contains( int value );
    // returns true if v is ≥ lower bound and ≤ upper bound,
    // and false otherwise

  public boolean overlaps( Range other );
    // returns true if the receiver contains at least
    // one value in common with other, and false otherwise

  public int size();
    // returns the number of integers in the range
}
```

```java
public class Disk
{
  private int   x;
  private int   y;
  private int   radius;
  private Color color;

  public Disk( int x, int y, int r )
  {
    this.x = x;
    this.y = y;
    radius = r;
    color  = Color.blue;
  }

  public void paint( Graphics g )
  {
    g.setColor( color );
    g.fillOval( x, y, radius*2, radius*2 );
  }

  protected void moveBy( int deltaX, int deltaY )
  {
    x += deltaX;
    y += deltaY;
  }
}
```

```java
import java.awt.Color;
import java.awt.Graphics;

public class Ball extends Disk
{
  private int deltaX;
  private int deltaY;

  public Ball( int x, int y, int r, int dx, int dy )
  {
    super( x, y, r );
    deltaX = dx;
    deltaY = dy;
  }

  public void move()
  {
    moveBy( deltaX, deltaY );
  }

  protected void reverseDeltaX()
  {
    deltaX *= -1;
  }

  protected void reverseDeltaY()
  {
    deltaY *= -1;
  }
}
```

```java
import java.awt.Frame;

public class BoundedBall extends Ball
{
  private Frame myWorld;

  public BoundedBall( int  x, int  y, int r,
                      int dx, int dy, Frame f )
  {
    super( x, y, r, dx, dy );
    myWorld = f;
  }

  public void move()
  {
    super.move();

    int maxWidth  = myWorld.getWidth();
    if (( x() < 0 ) || ( x() > maxWidth ))
      reverseDeltaX();

    int maxHeight = myWorld.getHeight();
    if (( y() < 0 ) || ( y() > maxHeight ))
      reverseDeltaY();
  }
}
```

```java
import java.awt.Frame;
import java.awt.Graphics;

public class BallWorldFrame extends Frame
{
  private static final int FrameWidth  = 600;
  private static final int FrameHeight = 400;
  private static final int Iterations  = 20000;

  private Ball ball;
  private int  counter;

  public BallWorldFrame()
  {
    super();

    setSize ( FrameWidth, FrameHeight );
    setTitle( "Ball World" );

    ball    = new Ball( 100, 100, 10, 10, 5 );
    counter = 0;
  }

  public void paint( Graphics g )
  {
    ball.paint( g );
    ball.move();
    counter++;

    try { Thread.sleep( 50 ); } catch (Exception e) {}

    if ( counter >= Iterations ) System.exit( 0 );

    repaint();
  }
}
```