Computer Science II - Test 2

Question 1. (15 points) The MouseListener interface requires methods for mouseClicked, mouseEntered, mouseExited, mousePressed, and mouseReleased. Java’s MouseAdapter implements MouseListener, but in MouseAdapter’s implementation none of these methods do anything. In the PinBallGame, a MouseKeeper inner-class extends MouseAdapter and overrides the mousePressed and mouseReleased methods.

a) Explain the reason why Java provides the MouseAdapter class.

b) Which of the types of inheritance (Specialization, Specification, Extension, Combination, Limitation, or Construction) does MouseKeeper characterize?

c) MouseAdapter is not an abstract class. You cannot make objects of abstract classes, but are forced instead to make subclasses from them via inheritance. Would it make sense for MouseAdapter to be defined as an abstract class? (Briefly justify your answer.)

Question 2. (15 points) A polymorphic variable is declared as maintaining a value of one type, but in fact holds a value from another type.

a) In the PinBallGame code, what is an example of a polymorphic variable? (be sure to clearly indicate the class, method, and name of the polymorphic variable)

b) Explain how the polymorphic characteristic of this polymorphic variable was created.
Question 3. (15 points) Given the class definitions shown in the box below, which of the following statements are legitimate in Java and why?

```java
interface ShapeInfo {
    public double area();
} // end interface ShapeCharacteristics

class Shape {
    protected int x;
    protected int y;

    public Shape(int inX, int inY) {
        x = inX;
        y = inY;
    }

    public String describe () {
        return "unknown shape";
    }

    public boolean equals ( Object arg ) {
        if (arg instanceof Shape) {
            Shape argS = (Shape) arg;
            if (x == argS.x && y == argS.y)
                return true;
        } // end if
        return false;
    }
} // end class Shape

class Square extends Shape implements ShapeInfo {
    protected int side;

    public Square(int inX, int inY, int inSide) {
        super(inX, inY);
        side = inSide;
    }

    public String describe () {
        return "square with side " + side;
    }

    public double area() {
        return (double) side * side;
    }
} // end class Square

class Circle extends Shape implements ShapeInfo {
    protected int radius;

    public Circle (int inX, int inY, int inRadius) {
        super(inX, inY);
        radius = inRadius;
    }

    public double area() {
        return (double) Math.PI * radius * radius;
    }
} // end class Circle
```

a) Shape s = new Square(6, 8, 3);
b) Circle c = new Square(10, 12, 4);
c) Square s = new Shape(9, 5);
d) Shape s = new Circle(8, 10);
e) ShapeInfo i = new Square(3, 4, 5);

Question 4. (10 points) What would be printed by the following code?

```java
Square s = new Square(2, 3, 4);
Square sqr = new Square(4, 5, 6);
System.out.println( s.describe() + " and " + sqr.describe() );

s = sqr;
System.out.println( s.describe() + " and " + sqr.describe() );

Circle c = new Circle(1, 2, 3);
System.out.println( c.describe() );
```
Question 5. (10 points) Consider the equals method of the Shape class in question 4.

a) Why is an equals method even necessary for comparing two Shape objects?

b) Write an equals method for the Circle class. We want two circles to be equal if they have the same values of x, y, and radius.

Question 6. (35 points) Attached is the refactored version 3 PinBallGame code discussed in class. **Complete one of the following questions.** For the question you pick, you do not have to completely rewrite classes. Instead, you may:
- write additional code on the existing code, and/or
- write additional code for the classes on blank paper, indicate in a comment for each block of code whether it replaces existing code or is added to existing code, and where the new code would go in the current code.

**Complete either part a or part b, but not both!**

a) Currently balls do not test to see if they intersect with other balls. We could support this modification by making PinBall implement the PinBallTarget interface, and adding balls to the list of targets as well as the list of balls. Modify the code to support balls bouncing off each other.

b) Develop a “paddle” target object. When a user clicks the mouse over the paddle, the paddle should move back and forth (perhaps only once). If a paddle encounters a ball, the ball is reflected off the paddle. Additionally, you should include code that adds a paddle to the pallet.
**PinBallGameDriver.java (Budd’s Version 3 Refactored):**

```java
public class PinBallGameDriver {
    public static void main( String[] args ) {
        PinBallGame.world = new PinBallGame();
        PinBallGame.world.show();
    } // end main
}  // end PinBallGameDriver
```

**PinBallGame.java:**

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

public class PinBallGame extends Frame {
    public static final int   FrameWidth = 400;
    public static final int   FrameHeight = 400;
    public static PinBallGame world;

    private int    score;
    private Label  scoreLabel;
    private Vector balls;
    private Vector targets;
    
    // ------------------------- PUBLIC INTERFACE -------------------------

    public PinBallGame() {
        setTitle( "Pin Ball Construction Kit" );
        setSize ( FrameWidth, FrameHeight );
        balls = new Vector();
        score = 0;
        scoreLabel = new Label( "score: 0" );
        add( "North", scoreLabel );
        addMouseListener( new MouseKeeper() );
        targets = new Vector();
        buildWalls();
        buildPalette();
    }

    synchronized public void addScore( int value ) {
        score = score + value;
        scoreLabel.setText( "score = " + score );
    }

    public Vector targets() {
        return targets;
    }

    // ------------------------- HELPER METHODS -------------------------

    protected void paintLauncher( Graphics g ) {
        g.setColor (Color.white);
        g.fillRect (FrameWidth-40, FrameHeight-40, 30, 30);
        g.setColor (Color.red);
        g.fillOval (FrameWidth-40, FrameHeight-40, 30, 30);
    }

    protected void paintBalls( Graphics g ) {
        for (int i = 0; i < balls.size(); i++)   {
            PinBall aBall = (PinBall) balls.elementAt( i );
            aBall.paint(g);
        }
    }

    public void paint( Graphics g ) {
        paintLauncher( g );
        paintBalls   ( g );
        paintTargets ( g );
    }
```

Fall 2005                                                                                                  Name: ______________________
protected void paintTargets( Graphics g ) {
    for (int j = 0; j < targets.size(); j++)
    {
        PinBallTarget target = (PinBallTarget) targets.elementAt( j );
        target.paint(g);
    }
}

protected void buildWalls() {
    targets.addElement(new Wall(50, 50, 2, 350));
    targets.addElement(new Wall(50, 50, 340, 2));
    targets.addElement(new Wall(390, 50, 2, 380));
}

protected void buildPalette() {
    targets.addElement(new Hole(30, 100));
    targets.addElement(new Peg(30, 140, 100));
    targets.addElement(new Peg(30, 180, 200));
    targets.addElement(new ScorePad(30, 220, 100));
    targets.addElement(new ScorePad(30, 260, 200));
    targets.addElement(new Spring(15, 300));
    targets.addElement(new Wall(30, 340, 2, 15));
}

// ------------------------- INNER CLASSES -------------------------

private class MouseKeeper extends MouseAdapter {
    private PinBallTarget element;

    public void mousePressed( MouseEvent e ) {
        element = null;
        int x = e.getX();
        int y = e.getY();

        if ( (x > FrameWidth-40) && (y > FrameHeight -40) ) {
            PinBall newBall = new PinBall( e.getX(), e.getY() );
            balls.addElement( newBall );
            Thread newThread = new PinBallThread( newBall, PinBallGame.this );
            newThread.start();
        } else if ( x < 40 ) {
            switch ( y / 40 ) {
                case 2: element = new Hole(0, 0); break;
                case 3: element = new Peg(0, 0, 100); break;
                case 4: element = new Peg(0, 0, 200); break;
                case 5: element = new ScorePad(0, 0, 100); break;
                case 6: element = new ScorePad(0, 0, 200); break;
                case 7: element = new Spring(0, 0); break;
                case 8: element = new Wall(0, 0, 2, 15); break;
            } // end switch
        } // end if
    } // end mousePressed

    public void mouseReleased( MouseEvent e ) {
        int x = e.getX();
        int y = e.getY();

        if ( (element != null) && (x > 50) ) {
            element.moveTo(x, y);
            targets.addElement (element);
            repaint();
        } // end if
    } // end mouseReleased

    } // end inner-class MouseKeeper
}

// end class PinBallGame

PinBallThread.java:
import java.util.Vector;

public class PinBallThread extends Thread {
    private PinBall theBall;
    private PinBallGame theWorld;

public PinBallThread( PinBall aBall, PinBallGame aWorld ) {
    theBall = aBall;
    theWorld = aWorld;
}

public void run() {
    while ( theBall.y() < theWorld.getHeight() ) {
        theBall.move();
        Vector targets = theWorld.targets();
        for ( int j = 0; j < targets.size(); j++ ) {
            PinBallTarget target = (PinBallTarget) targets.elementAt( j );
            if ( target.intersects( theBall ) )
                target.hitBy( theBall );
        } // end for
        theWorld.repaint();
        try { sleep( 10 ); }
        catch ( InterruptedException e ) { System.exit( 0 ); }
    } // end while
} // end run
} // end class PinBallThread

PinBallTarget.java:
import java.awt.Graphics;

public interface PinBallTarget {
    public boolean intersects( PinBall aBall );
    public void moveTo( int x, int y );
    public void paint( Graphics g );
    public void hitBy( PinBall aBall );
} // end interface PinBallTarget

Hole.java:
import java.awt.*;

public class Hole extends Ball implements PinBallTarget {
    public Hole( int x, int y ) {
        super( x, y, 12 );
        setColor( Color.black );
    }

    public boolean intersects( PinBall aBall ) {
        return region().intersects( aBall.region() );
    }

    public void moveTo( int x, int y ) {
        super.moveTo( x, y );
    }

    public void hitBy( PinBall aBall ) {
        aBall.moveTo( 0, PinBallGame.FrameHeight + 30 );
        aBall.setMotion( 0, 0 );
    }
} // end class Hole

ScorePad.java:
import java.awt.*;

public class ScorePad extends Hole {
    private int value;

    public ScorePad( int x, int y, int v ) {
        super( x, y );
        value = v;
        setColor( Color.red );
    }
public void hitBy(PinBall aBall) {
    PinBallGame.world.addScore(value);
}

public void paint(Graphics g) {
    g.setColor(color);
    g.drawOval(region.x, region.y, region.width, region.height);
    String s = "" + value;
    g.drawString(s, region.x, y()+2);
} // end class ScorePad

Peg.java:
import java.awt.*;

public class Peg extends ScorePad {
    private int state = 1;

    public Peg(int x, int y, int v) {
        super(x, y, v);
        setColor(Color.green);
    }

    public void paint(Graphics g) {
        if (state == 2) {
            setColor(Color.red);
            g.drawOval(region.x-3, region.y-3,
                region.width+6, region.height+6);
            state = 1;
        } else {
            setColor(Color.green);
            g.drawOval(region.x-2, region.y-2,
                region.width+4, region.height+4);
        } // end if
        super.paint(g);
    } // end paint

    public void hitBy(PinBall aBall) {
        super.hitBy(aBall);
        aBall.setMotion(-aBall.yMotion(), -aBall.xMotion());
        while (intersects(aBall))
            aBall.move();
        state = 2;
    } // end class Peg

Spring.java:
import java.awt.*;

public class Spring implements PinBallTarget {
    private Rectangle pad;
    private int state;

    public Spring(int x, int y) {
        pad = new Rectangle(x, y, 30, 3);
        state = 1;
    }

    public void moveTo(int x, int y) {
        pad.setLocation(x, y);
    }

    public void paint(Graphics g) {
        int x = pad.x;
        int y = pad.y;
        g.setColor(Color.black);
        if (state == 1) {
            g.fillRect(x, y, pad.width, pad.height);
            g.drawLine(x, y+3, x+30, y+5);
            g.drawLine(x+30, y+5, x, y+7);
Wall.java:
import java.awt.*;

public class Wall implements PinBallTarget {
    public Rectangle location;

    public Wall( int x, int y, int width, int height ) {
        location = new Rectangle( x, y, width, height );
    }

    public void moveTo( int x, int y ) {
        location.setLocation( x, y );
    }

    public void paint( Graphics g ) {
        g.setColor( Color.black );
        g.fillRect( location.x, location.y, location.width, location.height );
    }

    public boolean intersects( PinBall aBall ) {
        return location.intersects( aBall.region() );
    }

    public void hitBy( PinBall aBall ) {
        if ( (aBall.y() < location.y) ||
            (aBall.y() > (location.y + location.height)) )
            aBall.setMotion( aBall.xMotion(), -aBall.yMotion() );
        else
            aBall.setMotion( -aBall.xMotion(), aBall.yMotion() - 0.5 );
    }
}

Ball.java:
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;

public class Ball {
    private Rectangle location;
    private Color     color;

    public Ball( int x, int y, int r ) {
        location = new Rectangle( x-r, y-r, 2*r, 2*r );
        color = Color.blue;
    }

    public boolean intersects( PinBall aBall ) {
        return pad.intersects( aBall.region() );
    }

    public void hitBy( PinBall aBall ) {
        if ( aBall.yMotion() > 0 )
            aBall.setMotion( aBall.xMotion(), -aBall.yMotion() );
        aBall.setMotion( aBall.xMotion(), aBall.yMotion() - 0.5 );
        state = 2;
    }
}

Fall 2005
Name: ______________________
public class MovableBall extends Ball {
    private double dx;
    private double dy;

    public MovableBall( int x, int y, int r, double dx, double dy ) {
        super( x, y, r );
        this.dx = dx;
        this.dy = dy;
    }

    public void move() {
        region().translate( (int) dx, (int) dy );
    }

    protected void setMotion( double ndx, double ndy ) {
        dx = ndx;
        dy = ndy;
    }

    protected double xMotion() {
        return dx;
    }

    protected double yMotion() {
        return dy;
    }
} // end class MovableBall

CannonBall.java:
import java.awt.Color;

public class CannonBall extends MovableBall {
    public CannonBall (int sx, int sy, int r, double dx, double dy) {
        super(sx, sy, r, dx, dy);
    }
} // end class CannonBall
public void move () {
    setMotion( xMotion(), yMotion() + 0.3 );
    super.move();
}
} // end class CannonBall

PinBall.java:
public class PinBall extends CannonBall {
    public PinBall( int sx, int sy ) {
        super( sx, sy, 10, -2 + Math.random()/4, -15 );
    }
} // end class PinBall