**Part 1 - Written Answers**

**Juanjuan Xia**

**1. List all the examples of polymorphism that you can find. That is to say, statements where the datatype of a reference is not an exact match with the datatype of the object that it refers to.**

gw.add(new MyCircle(10, 10, 9));

gw.add(new MyCircle(25, 20, 12));

gw.add(new MyCircle(25, 20, 5));

gw.add(new MyRectangle(25, 25, 20, 15));

gw.add(new MyRectangle(5, 5, 3, 4));

gw.add(new MyRectangle(40, 0, 10, 10));

**2. In the containsPoint method of MyCircle, why are x and y in scope, even though they are not defined as instance variables in MyCircle.java?**

Because x and y are protected members in GridItem. At the same time, Mycircle.java is subclass for GridItem.

Protected members of a class could be accessed by methods in a subclass, and by methods in the same package as the class.

**3. Put a System.out.println statement into each containsPoint method (GridItem, MyCircle, and MyRectangle). Run the program and examine the output. You will notice that GridItem’s containsPoint method is never called. Why is this?**

gw.add(new MyCircle(10, 10, 9));

gw.add(new MyCircle(25, 20, 12));

gw.add(new MyCircle(25, 20, 5));

gw.add(new MyRectangle(25, 25, 20, 15));

gw.add(new MyRectangle(5, 5, 3, 4));

gw.add(new MyRectangle(40, 0, 10, 10));

These statements declare *item* as Griditem variable. They create MyCircle and MyRectangle objects and store the objects’ addresses in the *item* variable. MyCircle and MyRectangle class extend the GridItem class, and they override the containsPoint method.

In Java Virtual Machine, it is the object’s type that determines which method is called, not the variable’s type. In the case, the item variable references MyCircle and MyRectangle objects, so the MyCircle and MyRectangle classes’ version of the containsPoint methods are called.

**4. Notice that x and y are declared as ‘protected’ in GridItem. Change this to ‘private’ and recompile. How does the compiler respond?**

The compiler shows that x and y in MyCircle and in MyRectangle are errored because x and y have private access in GridItem.

**5. How could Abstract Methods have been used to make the code cleaner?**

Change the GridItem.java as following.

public **abstract** class GridItem {

protected int x;

protected int y;

public int getX() {return x;}

public void setX(int value) {x = value;}

public int getY() {return y;}

public void setY(int value) {y = value;}

**public abstract double getArea();**

**public abstract boolean containsPoint(int xValue, int yValue);**

}

**6. How might an Interface have been used to structure the classes?**

public interface Relatable {

double getArea();

boolean containsPoint(int xValue, int yValue);

}