

Answer all of the following questions. READ EACH QUESTION CAREFULLY. Answer each question in the space provided on these pages. Budget your time so you spend enough on the longer programming questions at the end. There are a total of **45** points.

Keep your answers short and to the point. Good luck.

Question 1. (4 points) Suppose we've declared and initialized the following variables

```
int k = 3;
int n = 10;
double x = 3.5;
```

What are the values of the following expressions?

- (a) $x + k / n$
- (b) $x * k / n$
- (c) $x + (\text{double}) k / n$
- (d) $x * (\text{double}) (k / n)$

Question 2. (3 points) Suppose we have two methods that compare two String objects, but in different ways.

```
public boolean one(String s, String t) {
    return s == t;
}

public boolean two(String s, String t) {
    return s.equals(t);
}
```

Do these methods do the same thing? If not, how do they differ?

Question 3. (2 points) Look at the following method heading

```
/** do something interesting */  
public void astonish(int k) { ... }
```

What does the keyword “void” mean when it appears in a method definition like this?

Question 4. (2 points) When discussing classes like `BankAccount`, we emphasized that instance variables like `accountName`, `balance`, and so forth should normally be declared to be `private`. For example,

```
public class BankAccount {  
    // instance variables  
    private String accountName;    // account holder's name  
    private double balance;        // account balance  
    ...  
}
```

Why is it normally good practice to declare the instance variables `private`?

Question 5. (2 points) Here is the definition of a very simple class.

```
public class Box {  
    // instance variable  
    private int val;  
  
    /** set the value of this Box to newVal */  
    public void setVal(int newVal) { this.val = newVal; }  
  
    /** return the value of this Box */  
    public int getVal( ) { return this.val; }  
}
```

Client code uses this class in the obvious way:

```
// create a new Box and play with it  
Box cardboard = new Box( );  
cardboard.setVal(5);  
cardboard.setVal(cardboard.getVal( ) + 17);
```

Now, a new programmer on the team decides it would be a great idea to add a constructor to class Box so an initial value can be supplied when a new Box is created.

```
/** construct new Box with given initial value */  
public Box (int initialVal) { this.val = initialVal; }
```

Unfortunately, the existing client code no longer works properly after this change – in fact, it doesn't even compile any more. Why not?

Question 6. (2 points) What is the difference between the statement

```
int xyz = 10;      /* statement 1 */
```

and the statement

```
xyz = 10;          /* statement 2 */
```

Explain precisely (but concisely).

Question 7. (6 points) Suppose we represent the location of a City with an object containing the city name, latitude, and longitude:

```
/** name and location of a single city */
class City {
    // instance variables
    private String name;        // city name
    private double latitude;    // city coordinates
    private double longitude;

    /** Construct a City record with the given name and location */
    public City(String name, double latitude, double longitude) {...}

    // ... other methods omitted to save space
}
```

Now, suppose we create a class to hold a list of City records:

```
class Cities {
    private ArrayList cities; // City records in this list

    /** Construct empty list of cities */
    public Cities {
        this.cities = new ArrayList( );
    }

    /** Add given city to this list */
    public void add(City city) {
        this.cities.add(city);
    }
}
```

On the next page, draw a picture showing the objects that are created when the following code is executed, and the relationships between the objects (i.e., which objects refer to which others).

```
Cities cityList = new Cities( );
cityList.add(new City("Seattle", 47.58, 122.33));
cityList.add(new City("Chicago", 41.83, 87.75));
cityList.add(new City("New York", 40.75, 74.50));
```

Question 7 (cont). Draw your picture here (code duplicated here for reference)

```
Cities cityList = new Cities( );  
cityList.add(new City("Seattle", 47.58, 122.33));  
cityList.add(new City("Chicago", 41.83, 87.75));  
cityList.add(new City("New York", 40.75, 74.50));
```

Question 8. (6 points) The Puck objects from homework 4 represented an object that could move around on a window, optionally drawing lines as it moves from place to place. Here are the instance variables from that class:

```
public class Puck {
    // instance variables
    private GWindow theWindow; // the drawing window for this puck
    private int x;              // current x,y location of this puck
    private int y;
    private Color color;        // current line color
    private boolean penDown;    // true if pen is currently down
    ...
}
```

Complete the following definition of a new method `moveHorizontal` for class `Puck`. When `moveHorizontal` is called, it should move the given distance relative to the Puck's current location, updating instance variables and drawing lines as appropriate.

Hint: Recall that class `Line` has two constructors that might be useful here:

```
Line(x1,y1,x2,y2)
Line(x1,y1,x2,y2,color)
```

New method for class `Puck`:

```
/** Move this Puck the given distance horizontally, drawing
 *  a line from the old to new location if appropriate.
 *  @param distance Horizontal distance to move from the
 *                  current location.
 */
public void moveHorizontal(int distance) {
```

```
}
```

Question 9. (18 points total) This is a longer question with several parts. You may find it useful to read the entire question first, before you fill in any of the answers.

Wanda & Wally's Widget Works (known as www) sells Widgets of various kinds. They would like to implement a computer database to store a record of their widget inventory.

Each Widget in the inventory will be represented by an object that contains a description of the widget, the widget part number, the unit cost of one widget of this kind, and the number of widgets of this kind that are currently in stock.

Here are the instance variables and constructor for class Widget.

```
/** Inventory record for a single widget */
public class Widget {
    // instance variables
    private String description; // widget description
    private int partNumber;    // widget part number
    private double unitCost;    // cost of each widget of this kind
    private int quantity;      // # of these widgets in inventory

    /** Construct new Widget with given description, etc. */
    public Widget(String description, int partNumber,
                  double unitCost, int quantity) {
        // (constructor body omitted to save space)
    }
}
```

(a) (4 points) For this part of the question, complete the toString method for class Widget.

```
/** return an appropriate string representation of this Widget */
public String toString() {
    
}
}
```

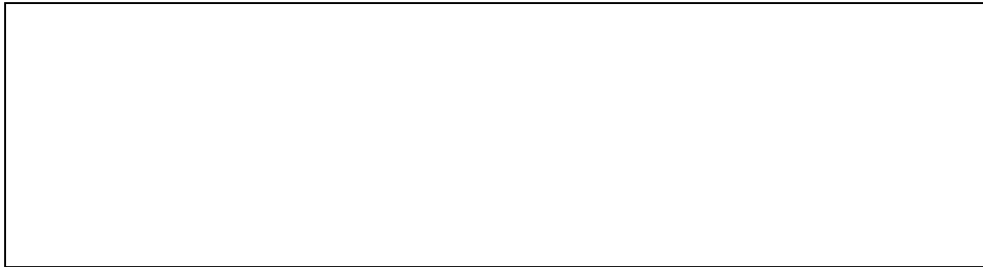
(b) (14 points) For this part of the question, implement three methods of class `Inventory`, which is a class that stores a list of `Widget` records. The three methods you are to implement are the constructor, `add` (which adds a new `Widget` record to the `Inventory`), and `mostExpensive` (which returns the description of the most expensive widget in the inventory).

For your reference, a complete description of class `Widget`, including methods to access fields in `Widget` objects, is given on the last page of this exam. Feel free to tear off that page so you don't have to flip back and forth while answering this part of the question.

Hint: Iterator objects include methods `hasNext()` and `next()`, which might be useful here.

```
/** List of Widget records */
public class Inventory {
    // instance variable
    private ArrayList widgets;

    /** Construct Inventory with empty list of widgets */
    public Inventory() {
```



```
}
```

```
/** Add the given Widget record to this Inventory list */
public void add(Widget w) {
```



```
}
```



```
/** Return the description of the most expensive Widget in this
 * Inventory.  If two or more Widgets have the same highest
 * unitCost, return the name of any one of them (i.e., you can
 * break ties however you like).  Return an empty string ("")
 * if the list is empty.
public String mostExpensive( ) {
```

```
}
```

Complete reference information about class Widget. Tear off if you wish.

```

/** Inventory record for a single widget */
public class Widget {
    // instance variables
    private String description; // widget description
    private int partNumber;    // widget part number
    private double unitCost;   // cost of each widget of this kind
    private int quantity;      // # of these widgets in inventory

    /** Construct new Widget with given description, etc. */
    public Widget(String description, int partNumber,
                  double unitCost, int quantity) { ... }

    /** Return the description of this Widget */
    public String getDescription( ) { ... }

    /** Return the part number of this Widget */
    public int getPartNumber( ) { ... }

    /** Return the unit cost of this Widget */
    public double getUnitCost( ) { ... }

    /** return number of these widgets in this inventory */
    public int getQuantity( ) { ... }

    /** return an appropriate string representation of this Widget */
    public String toString( ) { ... }
}

```

Instance variable for class Inventory.

```

/** List of Widget records */
public class Inventory {
    // instance variable
    private ArrayList widgets;
    ...
}

```