DO NOT SPEND MORE THAN 15 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later.

Before starting, make sure your test contains 10 pages. The last page are blank and can be used as scratch paper, but all pages must be turned in.

None of the Java programs or program segments should have syntax errors unless stated. If you think there is a syntax error or you do not understand what a question is asking, then please ask. Import statements may not always be shown.

Do not discuss this test with anyone until the test is handed back.
**PROBLEM 1:**  (Maximum impact: 8 points)

**Part A:** (4 points)
Complete the method `allDifferent` that returns true only if all three of the given numbers have the different values. For example, the call `allDifferent(3, 4, 1)` should return true; while the call `allDifferent(3, 4, 3)` should return false. You should not assume that any additional instance variables exist.

```java
public boolean allDifferent(int a, int b, int c) {
    // Implementation
}
```

**Part B:** (4 points)
Complete the method `max3` that takes three numbers and returns the one with the largest value (if more than one of the values are equally the largest, then return the first such value). Your implementation may not use any conditionals or loops and **must** call the static method `Math.max` at least once and use its result in determining the result of this function. You should not assume that any additional instance variables exist.

```java
public int max3(int a, int b, int c) {
    // Implementation
}
```
PROBLEM 2:  (Understanding NameSurfer: (12 points))

For each of the following questions below, please answer briefly in complete sentences.

1. In the NameSurfer program, is there ever a NameRecord created such that a call to getRank with a valid decade, i.e., between 0 and 11 inclusive, would cause an error in the program? Explain why or why not.

2. In the NameSurfer program, if the ranks of the names were expanded to include the top 2000 most popular names, instead of the top 1000 names, explain what one line of code would need to be changed and what one method it would affect.

3. In the NameSurfer program, if the data file provided were expanded to include ranking data from the 1800s, i.e., adding ten more decades of ranks, describe (but do not code) the parts of the code that would need to be changed.
PROBLEM 3 : (Highest Ranking: (14 points))

Part A: (6 points)
In the NameSurfer program, if the user were allowed to enter more than one name to search for and the results somehow showed all the names together on the same display, explain (but do not code) how to change the Canvas class to support this feature. Specifically, this change would mean that the Canvas's setName method would be given an ArrayList of NameRecords, rather than a single one. Describe the other changes that would be needed.

Part B: (8 points)
Given the code on the next page from the paintComponent method of the Canvas class of the NameSurfer program, modify it such that instead of coloring all bars in the graph the same color, it colors every bar which represents a decade of the name's highest rank the color java.awt.Color.CYAN and all others java.awt.Color.GREEN.

For example, for the following two names and their associated ranks:

Susan 156 185 219 121 10 4 4 28 96 240 429
Ralph 23 21 24 30 45 62 109 186 272 448 588

your revised version of the code should specially color both decades Susan was ranked 4 and the one decade Ralph was ranked 21.

Problem continued on next page ...
Dimension bounds = getSize();
for (int k = 0; k < myYears.size(); k++)
{
    // get the rank and relative amount to size
    int rank = myName.getRank(k);
    double percent = getPercent(rank, NameDatabase.MAX_RANK);

    // determine the size of the bar in the graph
    Dimension size = new Dimension(bounds.width / myYears.size(),
                                    (int)((bounds.height - 100) * percent))
    Point topLeft = new Point(k * size.width + 5,
                              bounds.height - size.height - 25);

    // zero indicates no data
    if (rank > 0)
    {
        // draw the bar
        pen.setColor(Color.GREEN);
        pen.fillRect(topLeft.x, topLeft.y, size.width, size.height);

        // label the rank
        pen.setColor(Color.BLACK);
        pen.drawString("" + myName.getRank(k), topLeft.x, topLeft.y);
    }
}
PROBLEM 4:  (Right on Target: (23 points))

Part A: (8 points)
Write the render method of a sub-class of Mover such that draws itself as a number of concentric ovals centered at the same position. They should be drawn from largest to smallest so that they are all visible and look like an archery target such that the distance between each is equal and each oval is brighter than the one drawn before it. The number of ovals drawn is passed as a parameter to the constructor as numRings.

For example, a target of three ovals should be drawn such that largest oval’s size is the same as that given for the entire target, the inner circle’s size is two-thirds of that given for the target, and the center circle’s size is one-third of that given for the target. Your render method should work for any target with a positive number of ovals.

Complete the class declared below (declaring and initializing any needed instance variables).

```java
public class Target extends Mover
{

    public Target (Point center, Dimension size, Point velocity, Color color,
                    int numRings)
    {
        super(center, size, velocity, color);

    }

    public void render (Graphics pen)
    {

    }

    // update method not used

}  
```
Part B: (6 points)
Write the update method of a subclass of Mover such that it exactly follows one step behind another Mover, given as its leader. In other words, it should make the exact same moves as its leader, never gaining or losing distance from it.
Complete the class Follower started below.

```java
public class Follower extends Mover {

public Follower (Point center, Dimension size, Point velocity, Color color, Mover leader) {
    super(center, size, velocity, color);
}

public void update (Dimension bounds) {

    // render method not used

} // render method not used
```

Part C: (9 points)
Complete the method createMovers for a Factory subclass that adds a single leading BouncingSmiley and a numToCreate of Follower objects to the given canvas object, such that each follows the one before it (the first one should follow the single leading smiley). Additionally, each follower object created should be one one-hundredths smaller than the one before it, darker than the one before it, and tangent to the one before it in the x direction. Thus, when created, it should look like a horizontal line of ovals, headed by a smiley face and getting smaller and darker towards the tail.

Complete the method createMovers below:

```java
public void createMovers (Canvas canvas, int numToCreate)
{
    Dimension bounds = canvas.getSize();

    Mover leader = new BouncingSmiley(
        new Point(bounds.width / 2, bounds.height / 2),
        new Dimension(100, 100),
        new Point(3, 3),
        new Color(255, 0, 255));
    canvas.add(leader);
}
```
**PROBLEM 5** : *(Order doesn't matter: 18 points)*

Complete the method, `createMovers`, for a `Factory` subclass that reads a data file and adds each `BouncingBall` read to the given `canvas` object. Assume that the text file with shape information being read is in the format shown below. Data about each attribute of the shape is given on a separate line, thus a single shape’s data may span multiple lines. Each shape may define up to four attributes: center, size, velocity, or color. If there is no definition of the attribute’s value in the data file, then it should be given a random value within reasonable bounds (i.e., within the size of the canvas for the center, between 0 and 255 for the color, and between the declared constants for size and velocity).

The attributes can be given in any order or not at all. If the first word on the line (disregarding whitespace) is an attribute name, it will be followed by the appropriate number of integers to construct that attribute (two each for `Point` or `Dimension` objects and three for `Color` objects). If the first word on the line (disregarding whitespace) is `shape`, then it will be followed some time later (either on the same line, or another line) by a matching word `epahs` to denote the end of the shape’s definition.

A sample data file representing four different shapes is shown below:

```plaintext
shape
  center 100 100
  size 100 100
  velocity 5 5
  color 255 0 255
epahs
shape
  center 100 200
  color 255 0 0
epahs
shape epahs
shape
  size 20 20
  velocity 7 -3
  center 300 200
epahs
```

The following constants are defined for your use:

```java
final int MIN_SIZE = 10;
final int MAX_SIZE = 50;
final int MAX_VELOCITY = 10;
final int MIN_VELOCITY = -MAX_VELOCITY;
final String SHAPE_BEGIN = "shape";
final String SHAPE_END = "epahs";
final String CENTER = "center";
final String SIZE = "size";
final String VELOCITY = "velocity";
final String COLOR = "color";
```
Complete the method `createMovers` below:

```java
public void createMovers (Canvas canvas, Scanner input) {
}
```
Throughout this test, assume that the following classes and methods are available. These classes are taken
directly from the material used in class. There should be no methods you have never seen before here.

### Point

```java
public class Point {
    // coordinates
    public int x;
    public int y;

    // Constructs and initializes a point at the
    // specified (x,y) location.
    public Point (int x, int y)

    // Returns distance from this point to given one
    public int distance (Point other)

    // Adds dx and dy to this point’s coordinates
    public void translate (int dx, int dy)
}
```

### Dimension

```java
public class Dimension {
    // lengths
    public int width;
    public int height;

    // Constructs and initializes a dimension
    // with the specified (w, h) lengths.
    public Dimension (int w, int h)
}
```

### Color

```java
public class Color {
    // Constructs and initializes a color with
    // the specified (r, g, b) components.
    public Color (int red, int green, int blue)

    // Returns the red component
    public int getRed ()

    // Returns the green component
    public int getGreen ()

    // Returns the blue component
    public int getBlue ()

    // Returns a color that is brighter than
    // this one
    public Color brighter ()

    // Returns a color that is darker than
    // this one
    public Color darker ()
}
```

### ArrayList

```java
public class ArrayList {
    // Constructs an empty list
    public ArrayList ()

    // Returns the number of elements in this list.
    public int size ()

    // Searches for the first occurrence of the given
    // argument, returns -1 if not found
    public int index0f (Object itemm)

    // Returns element at index in this list.
    public Object get (int index)

    // Appends specified element to end of this list.
    public boolean add (Object o)
}
```

```java
// To create an ArrayList of String objects:
ArrayList<String> things = new ArrayList<String>();
```

### Scanner

```java
public class Scanner {
    // Create Scanner for that reads data from a string.
    public Scanner (String str)

    // Check if more items are available
    public boolean hasNext ()

    // Get next delimited item as a string
    public String next ()

    // Get next delimited item as an integer value
    public int nextInt ()
}
```

```java
// typically used in a while loop:
Scanner input = new Scanner("a b c");
while (input.hasNext())
{
    String letter = input.next();
    // do something with letter
}
```
Random

public class Random {
    // Creates a new random number generator.
    public Random ()

    // Returns pseudorandom, uniformly distributed,
    // int value between 0 (inclusive) and the
    // specified value (exclusive)
    public int nextInt (int n)
}

Math

public class Math {
    // Returns the greater of the two given values
    public int max (int a, int b)
    // Returns the lesser of the two given values
    public int min (int a, int b)
    // Returns the sin of the given angle
    public double sin (double angleInRadians)
    // Returns the sin of the given angle
    public double cos (double angleInRadians)
}

// do not create a Math object to call Math methods:
int greater = Math.max(10, 2);

String

public class String {
    // Returns true if the given string contains
    // the exact same characters as this string
    public boolean equals (String other)

    // Returns true if the given string contains
    // the same characters as this string regardless
    // of whether they are upper- or lower-case
    public boolean equalsIgnoreCase (String other)
}