First Exam
Computer Programming 326
Dr. St. John
Lehman College
City University of New York
Thursday, 7 October 2010

NAME (Printed) _____
NAME (Signed) _____
E-mail _____

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes.
- When taking the exam, you may have with you pens or pencils, and an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- You may not use a computer or calculator.
- All books and bags must be left at the front of the classroom during this exam.
- **Do not open this exams until instructed to do so.**

Question 1	
Question 2	
Question 3	
Question 4	
Question 5	
Question 6	
Question 7	
Question 8	
Question 9	
Question 10	
TOTAL	

1. True or False:

   (a) ___ An algorithm is a a set of directions for solving a problem.

   (b) ___ You can use the "$*$" symbol to indicate the concatenation of two strings.

   (c) ___ The `switch` and `while` statements are examples of branching statements.

   (d) ___ Arrays always start with an index of 1.

   (e) ___ Everything that can be done with `for` statement can be done with a `while` statement.

   (f) ___ In Java, all variables are local.

   (g) ___ The operators = and == perform the same action.

   (h) ___ Arrays cannot be parameters (inputs) to a method.

   (i) ___ Constructors must have parameters.

   (j) ___ A static variable is shared by all the objects of its class.

2. Write the Java code that declares and instantiates:

   (a) a integer `count` that holds the number 1:

   (b) a double `pi` which is `3.1459`:

   (c) a string `lastName` that holds your name:

   (d) an array `friends` of 10 `Person` objects:

   (e) that converts the string `inputString` into an integer, `x`:

3. What is the output of the following code fragments:

(a) **Output:**

```
int numtimes = 5;
while ( numtimes >= 2 )
{
    System.out.print("Hi!");
    numtimes--;
}
System.out.print("Bye!");
```

(b) **Output:**

```
boolean done = false;
int total = 5;
while ( !done )
{
    if ( total > 4 )
    {
        done = true;
    }
    total = total*2;
}
System.out.println(total);
```

(c) **Output:**

```
int i, j;
for ( i = 3 ; i > 0 ; i--)
{
    for ( j = 0 ; j < i ; j++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

(d) **Output:**

```
int i, j;
for ( i = 0 ; i < 4 ; i++)
{
    for ( j = 0 ; j < 4 ; j++)
    {
        if ( i+j >  3 )
        {
            System.out.print("+");
        }
        else
        {
            System.out.print("-");
        }
    }
    System.out.println();
}
```

4. What is the output when the code is run?

(a) **Output:**

```
char[] letters = {'a','z','e','b','r','a'}
int m = 0;
for (int i = 1; i < letters.length; i++) {
  if ( letters[i]  > letters[m] ) {
    m = i;
  }
  System.out.println(i + ": " + letters[i]
                     + " " + letters[m]);
}
```

(b) **Output:**

```
double[] s = {5,1.2, 0.3, 5.89, 2, 0.1}
for (int i = 0; i < s.length-1; i++) {
  for (int j=0; j < s.length-1; j++) {
    if ( s[j]  > s [j+1] ) {
      double tmp = s[j];
      s[j] = s[j+1];
      s[j+1] = tmp;
    }
    System.out.print(s[j] +  " ");
  }
  System.out.println();
}
```

5. Declare and instantiate the following arrays:

(a) an array, `scores`, that holds ten integers:

(b) an array, `even`, that holds the even numbers from 0 to 10:

(c) a 2 dimensional array, `board`, of 3 rows by 4 columns of characters:

6. Assume the following class definition:

```
public class Mystery {
    public int[] numbers;
    public String message;
    public Mystery()
    { numbers = {1,2,3}; message = "Hello"; }
    public String toString()
    {   return(numbers.length+" "+message); }
    public void query()
    {   int i;
        System.out.print(message);
        for ( i = 0 ; i < numbers.length ; i++ )
            System.out.print(numbers[i] + " ");
        System.out.println();
    }
}
```

and the following code has been executed:

```
Mystery first = new Mystery();
Mystery second;
first.numbers[0] = 4;
first.message = "Hi";
second = new Mystery();
second.numbers[1] = 2*first.numbers[1];
```

What is the output from the following statements?

(a) `System.out.print(first.message);`

Output:

(b) `first.query();`

Output:

(c) `System.out.print(first);`

Output:

(d) `second.query();`

Output:

(e) `System.out.print(second);`

Output:

7. Examine the class below and answer the following:

   (a) What is the output of this program?

   (b) How many times is the first `getAverage()` method called?

   (c) How many times is the last `getAverage()` method called?

   (d) In your own words, describe how the compiler decides which of the `getAverage()` methods to call:

```
public class Overload
{
    public static void main(String[] args)    {
        double average1 = Overload.getAverage(40.0, 50.0);
        double average2 = Overload.getAverage(1.0, 2.0, 3.0);
        char   average3 = Overload.getAverage('a', 'c');
        System.out.println("average1 = " + average1);
        System.out.println("average2 = " + average2);
        System.out.println("average3 = " + average3);
    }
    public static double getAverage(double first, double second)    {
        return (first + second) / 2.0;
    }
    public static double getAverage(double first, double second,
                                    double third)  {
        return (first + second + third) / 3.0;
    }
    public static char getAverage(char first, char second)  {
        return (char)(((int)first + (int)second) / 2);
    }
}
```

8. (a) Write a method that returns the maximum of two integers:

```
public static int max(int first, int last) {

}
```

(b) Write a method that takes an array of real numbers and returns the average number:

```
public static double average( double[] in) {

}
```

(c) Write a method that interchanges two elements in an array of `String` at the given indices

```
public static void swap( String[] a, int first, int last) {

}
```

9. Fill in the following class that holds information about a student. Each of the methods of the class is proceeded by a comment that explains what the method should do. Fill in each method with the appropriate code:

```
public class Student
{
    public String name;    //Student's name
    public int studentID; //Student ID number
    public int numCredits;   //Number of credits completed
    public String[] currentCourses;  //Names of current classes

    public Student(String t, int n, int m, String[] c) {
       name = t;  studentID = n; numCredits = m; currentCourses = c;
    }
   /* Returns the student ID number.  */
   public int getID() {


   }
   /* Set the student ID number.  */
   public void setID(int n){


   }
   /* Returns all the information about the student (including the current courses): */
   public String toString() {









    }
    /* Returns true if taken more than 60 credits, otherwise returns false*/
   public boolean isUpperClassman() {




   }
}
```

10. Create a new class called `Lighten` that implements `ActionListener`. Your class has access to a global variable `picture` of the `Picture` class and `pictureFrame` of class `PictureFrame`. Your class should refresh the `pictureFrame` with a lighter version of the picture.

```
class Lighten implements ActionListener {
  public void actionPerformed (ActionEvent e) {









  }
}
```

The API for the `Picture` and `Pixel` classes are included on the next page:

Class Picture

java.lang.Object
  Picture

public class **Picture**
extends Object

The Picture class stores a two-dimensional image. This Picture is comprised of pixels (see Pixel) and is displayed on-screen through the use of a PictureFrame.

Author:
Richard Wicentowski

Constructor Summary

Picture(int width, int height) Create a Picture with the specified width and height.
Picture(Picture source) Create a Picture by copying the contents of another (non-null) Picture.
Picture(String filename) Create a Picture from a file.

Method Summary

BufferedImage	**getBufferedImage**() Returns the BufferedImage underlying the Picture.
int	**getHeight**() Return the height of the Picture.
Pixel	**getPixel**(int x, int y) Return the Pixel stored at this location.
int	**getWidth**() Return the width of the Picture.
Pixel	**setPixel**(int x, int y, Pixel pixel) Set the location (x, y) to a new Pixel value.
void	**updateImage**() Flush changes to the underlying image.

Class Pixel

java.lang.Object
  Pixel

public class **Pixel**
extends Object

The Pixel is the underlying data-structure in a Picture. Each Pixel represents a color, stored as separate integer values for red, green and blue. Each value is between 0 and 255. Collectively, these RGB (Red-Green-Blue) values form the color of the Pixel.

Author:
Richard Wicentowski

Constructor Summary

Pixel(int r, int g, int b) Create a Pixel with the specified RGB values.

Method Summary

boolean	**equals**(Object other) Determine if this Pixel is equal to another Object.
int	**getBlue**() Returns the blue component of the Pixel.
int[]	**getComponents**() Returns the RGB components of this Pixel as a 3-place int array.
int	**getGreen**() Returns the green component of the Pixel.
int	**getRed**() Returns the red component of the Pixel.
void	**setBlue**(int blue) Set the blue component of the Pixel.
void	**setGreen**(int green) Set the green component of the Pixel.
void	**setRed**(int red) Set the red component of the Pixel.
String	**toString**() Return the String representation of this Pixel as a triple (R, G, B).