Second Exam
Computer Programming 326
Dr. St. John
Lehman College
City University of New York
Thursday, 11 November 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 interface has no constructors or method definitions.

   (b) ___ A class that implements an interface defines each specific method.

   (c) ___ Polymorphism allows different objects to use different actions for the same name.

   (d) ___ A class can only implement one interface.

   (e) ___ An abstract method has no body.

   (f) ___ An exception signals an unusual event during execution.

   (g) ___ If no exception occurs within the try block, the catch blocks are ignored.

   (h) ___ A finally block always executes.

   (i) ___ The stream name and file name must be the same.

   (j) ___A flow of data into or out of a program is called a river.

2. (a) Define overloading and give an example:

   (b) Define overriding and give an example:

   (c) What is the difference between overloading and overriding?

3. (a) Write the statement that will throw a new exception with the message "Help!":

   (b) Write the statement that catches an IO exception:

   (c) Write the first line of a java class called `Employee` than extends the class `Person`:

   (d) Write the first line of a java class called `MyApplet` than implements the interface `JApplet`:

4. Given the classes:

```
public class Mystery
{
    private int x;
    private String s;
    Mystery() {  x = -10; s = "Hi Mom!";  }
    Mystery(int i) {  x = i; s = "Bye!";  }
    public String toString() {  return(x + " " + s);  }
    public int get() {  return (x);  }
}
public class Enigma extends Mystery {
    private int code;
    Enigma() {  super(); code = 42;  }
    Enigma(int i, int j){  super(i); code = j;  }
    public String toString() {  return(super.toString() + " " + code);  }
}
```

Assume the following code has been executed:

```
Mystery first = new Mystery();
Enigma second = new Enigma();
Enigma third = new Enigma(5,10);
```

What is the output from the following statements?

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

Output:

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

Output:

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

Output:

(d) `System.out.print(first.get());`

Output:

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

Output:

5. Given the following program:

```
import java.util.*;
public class MessageTooLongExceptionDemo {
    private static String inputLine;
    private static int lineLength;
    public static void main(String[] args)    {
        Scanner keyboard = new Scanner(System.in);
        try {
            keyboard = new Scanner(System.in);
            System.out.println("\nEnter a line of text up to 20 characters long.");
            inputLine = keyboard.nextLine();
            lineLength = inputLine.length();
            if (lineLength > 20) // Uses default message
                throw new Exception("Message too long!");
            System.out.print("You entered " + lineLength
                 + " characters, " + "which is an acceptable length.");
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
        System.out.println("Bye!");
    }
}
```

(a) What is the output of the program if the user enters: `Hi!`

(b) What is the output of the program if the user enters: `12345678901234567890123456789O`

6. Assuming `iStream` has been instantiated and linked to the file shown, what is the output of the following?

(a)

```
while (iStream.hasNextLine())
{
  String line = iStream.nextLine();
  System.out.println(line);
  System.out.println(line);
}
```

File:

```
1 Fish
2 Fish
Red Fish
Blue Fish
```

Output:

(b)

```
while (iStream.hasNextInt())
{
  int line = iStream.nextInt();
  System.out.println(line);
  System.out.println(line);
}
```

File:

```
1 Fish
2 Fish
Red Fish
Blue Fish
```

Output:

(c)

```
int count = 0;
while (iStream.hasNextLine())
{
  String line = iStream.nextLine();
  count++;
  System.out.println(count+" "+line);
}
```

File:

```
1 Fish
2 Fish
Red Fish
Blue Fish
```

Output:

(d)

```
int count = 0;
while (iStream.hasNextLine())
{
  String line = iStream.nextLine();
  count= line.length();
  System.out.println(count);
}
```

File:

```
1 Fish
2 Fish
Red Fish
Blue Fish
```

Output:

7. (a) Write a method that takes as input a file stream and a message, and prints the message to the file stream

```
public static void messageToFile( PrintWriter out, String message ) {




}
```

(b) Write a method that takes a file as input, and returns true if the file exists and can be written to. Otherwise, it returns false:

```
public static boolean isModifiable( File f ) {




}
```

(c) Write a method that takes a file name, and returns the number of characters in the file. (Note: you should open the file and handle any exceptions.)

```
public static int charCount( String fileName ) {




}
```

8. Create a class `SchoolKid` that is the base class for children at a school. It should have attributes for the childs name and age, the name of the childs teacher, and a greeting. It should have a default constructor, and appropriate accessor and mutator methods for each of the attributes.

9. Derive a class `ExaggeratingKid` from `SchoolKid`, as described above. The new class should override the accessor method for the age, reporting the actual age plus 2. It also should override the accessor for the greeting, returning the childs greeting concatenated with the words I am the best.

10. Write a **complete** program that will ask the user for the name of a file and will print out the number of lines in the file.