CSCI 1227 Midterm Test

Thursday, July 26, 2012

Name: _____

Student #: _____

General Instructions

Read and follow all directions carefully.

When writing programs or program segments, use the conventions used in the text for identifiers and indentation. Print clearly so that the grader can tell that the conventions are being followed.

You may write **S.o.p** for **System.out.print**, and **S.o.pln** for **System.out.println**. Any other abbreviations you use should be noted in the margin.

Note: you should not need more space than is provided. If it looks like you won't have enuf space to complete the question, then you should maybe consider the possibility that you're doing the wrong thing.

At **no point** do I ask you to write a complete program.

There are 76 points available. You have 90 minutes to complete it. Good luck.

Question	Score	Out of
1		6
2		8
3		4
4		10
5		6
6		4
7		4
8		10
9		6
10		12
11		6
Total		**76**

1. **[6]** Declare and initialize (on one line) the following variables and objects. You do not need to declare program constants.

   a) A matrix to hold a year's worth (365 days) of daily sales data for up to 100 stores.

   b) An array to hold up to 200 boolean values.

   boolean[] arr = new boolean[200];

   c) An array holding the words "Oh" and "boy". *Do not use assignment statements.*

   String[] words = new String{"Oh", "Boy"};

2. **[8]** Create and initialize an array of Person objects according to the pseudo-code given. (The definition of the Person class can be found on the **extra code page**.)

```
Scanner kbd = new Scanner(System.in);
System.out.print("How many people are invited? ");
int numPeople = kbd.nextInt();      kbd.nextLine();

// create an array the right size to hold the invited people

Person[] people = new Person[numPeople];


// ask for each Person's name and save that Person in the array
for (int i = 0; i < numPeople; ++i) {
    System.out.print("Enter a person's name: ");
    String name = kbd.nextLine();
    People[i] = new Person(name);
}
```

3. **[4]** Suppose we have given the following declarations (in two different files).

```
public class Manifest {
    public int      numEntries;  // NOTE: public, not private
    public String[] entry;       // NOTE: public, not private
};

Manifest[] arr = new Manifest[100];
/* arr initialized and filled in - details hidden */
```

Give the **data type** (in Java) of each of the following expressions.

a) arr Manifest[] _____

b) arr[5] Manifest _____

c) arr[5].entry String[] _____

d) arr[5].entry[4] String_____

4. **[10]** Suppose we have created the following matrix of weekly cookie sales data for five Brownie troops (eight weeks of sales each). (Assume that constants TROOPS = 5 and WEEKS = 8 have been declared.)

sales

45.00	42.00	36.00	59.00	85.00	40.00	96.00	70.00
60.00	40.00	28.00	70.00	92.00	45.00	98.00	76.00
68.00	48.00	32.00	74.00	88.00	52.00	92.00	64.00
62.00	35.00	25.00	71.00	64.00	43.00	84.00	50.00
45.00	28.00	15.00	66.00	43.00	21.00	76.00	20.00

Write a code fragment to calculate and print the total sales for each week.

5. **[6]** We want to create a class to represent the exceptional condition of a negative number being given where it's not allowed. Show the definition of the class NoNegativesException. (The default message should be "No negatives!")

6. **[4]** Match the exception to the situation it arises in (fill the letters from the descriptions further down into the spaces provided below – not all descriptions will be used).

   i) ____ ArrayIndexOutOfBoundsException

   ii) ____ InputMismatchException

   iii) ____ NoSuchElementException

   iv) ____ NullPointerException

   a) Asked for a method that doesn't exist.

   b) Asked for more data when there is no more data.

   c) Asked for something from a variable that's not referring to any object.

   d) Requested an array element that doesn't exist.

   e) The data entered isn't suitable for the kind of data required.

   f) Told Java an object was of a specific type when it wasn't of that type.

7. **[4]** Write a method (printInReverse) to print all the elements of an array of integers in reverse order. (For example, the array below will be printed as `100 34 67`.)

67	34	100

```
private static void printInReverse(int[] arr) {
    for (int i = 0; i < arr.length; ++i) {
        sop(arr[arr.length - i - 1]);
    }
}

// public would be OK instead of private
```

8. **[10]** Write a *standard* equals method for a class called **Child** that inherits from a class called **Parent**. The **Child** class declares a single field (an int called **level**) but inherits several important fields from **Parent**.

9. **[6]** Consider the classes shown on the **extra code page**. Which methods does

   a) Student inherit (without overriding) from Person?

   getName

   setName

   OK if also say Person

   b) Student override from Person?

   writeOutput

   c) Student create new on its own?

   getStudentNumber

   sendBill

   OK if also say Student

10. **[12]** Multiple Choice: select the *best available* answer from the options shown.

    Declaring a class final (`public final class FinalClass {…}`) means that

    a) every class (static) variable in the class is constant.
    b) every instance and class variable in the class is constant.
    c) every instance variable in the class is constant.
    d) no class can extend it (it can have no subclasses).
    e) none of its methods can be over-written/over-ridden (in any subclass).

    If we create an array with 200 elements, those components will be numbered from

    a) 0 to 199.
    b) 0 to 200.
    c) 1 to 199.
    d) 1 to 200.
    e) 1 to 201.

What is the output of the following code?

```
int[] a = new int[10];
System.out.println(a[10]);
```

a) 0

b) 10

c) (nothing will be printed because the code will not compile)

d) (nothing will be printed, because the program will crash)

e) (we don't know what'll be printed, because the elements were not initialized)

If a class wants to invoke the version of a method (named `doThis()`) that was defined in its grandparent class but overwritten in its parent class,

a) it should invoke it using the command `doThis()`.

b) it should invoke it using the command `super.doThis()`.

c) it should invoke it using the command `super.super.doThis()`.

d) it should invoke it using the command `super(super(doThis()))`.

e) (it can't invoke that method directly)

If we give a program numbers as command line arguments (as in `java addArgs 5 10 8`), then

a) the numbers are all available in one String, and we need to use a Scanner object to read them as integers.

b) the numbers are available as integers in the array args.

c) the numbers are available as integers or Strings, depending on how the main method was declared.

d) the numbers are available as Strings in the array args.

e) those numbers are not available to the program.

Suppose we have the following (*correct!*) code:

```
double[] arr = new double[]{10.1, 20.3, 30,7};
doThis(arr);
```

Which of the following is a suitable header line for the definition of doThis?

a) public static double[] doThis()

b) public static double[] doThis(double a, double b, double c)

c) public static void doThis(double a, double b, double c)

d) public static void doThis(double x)

e) public static void doThis(double[] a)

If a try block has many associated catch blocks, then an exception thrown from that try block will be handled

a) by the first catch block.
b) by the catch block with the best matching parameter.
c) by the first catch block with a parameter that matches the thrown exception.
d) by the last catch block with a parameter that matches the thrown exception.
e) (you cannot have multiple catch blocks associated with a single try block)

If the method `method` may result in the exception `MyException` being thrown, then the method header should be

a) `public void method() catch MyException`
b) `public void method() throw MyException`
c) `public void method() throw new MyException()`
d) `public void method() throws MyException`
e) `public void method() throws new MyException()`

Given the (partial) class definitions on the **extra code page**, which of the following commands gives a compile-time error?

a) Person a = new Student("Bill");
b) Person b = new Undergraduate("Emily");
c) Student c = new Person("Carol");
d) Student d = new Undergraduate("Denis");
e) (they *all* give compile-time errors)

The exception InputMismatchException can be imported from the library/package

a) exception.java.
b) java.exception.
c) java.io.
d) java.util.
e) util.exception.

The command to sort an array (of int values) named myNumbers would be

a) Arrays.sort(myNumbers)
b) myNumbers.sort()
c) sort(Arrays.myNumbers)
d) sort(myNumbers)
e) sort(new int[] myNumbers)

If we have a Person variable p, and we want to check to see if it holds a Student object, the way to do that is

a) if (p extends Student)

b) if (p instanceof Student)

c) if (Student extends Person)

d) if (Student instanceof Person)

e) if (Student throws Person)

11. **[6]** Consider the Student class shown on the **extra code page**. The constructor header has been provided, but the body is hidden. Complete the definition of the constructor below so that the Student's name is set to n and student number is set to num. ***(Important note: there is no default constructor for the Person class.)***

```
public Student(String n, int num)
{
    super(n);
    stuNo = num;




}
```

(This page is for your rough calculations. You may tear it off the exam booklet.)