Practice Questions for APS101[*]

Winter 2008, Hojjat Ghaderi

1 Java / OO

- What is `JVM`? Why `Java` is a more portable language than, for example, `C`?
- Name three forms of polymorphism in Java. Give an example for each.
- Explain the keywords `super, this, extends, private,static, throws` (what's their purpose or when/where they are used).
- What is Inheritance? What is overriding? What is overloading?
- Describe how Java compiler checks if the method call `o.m(10,"A")` is valid or not.
- Assuming `o` is declared as `Object`, describe how JVM figures out what method should be called when it reaches the `o.m(10,"A")` statement (explain different scenarios based on what might actually be in `o`).
- Also: Practice examples similar to those of `OO.java` from lecture 37 and 38 (array of Objects, casting to different classes, inheritance, instanceof, overriding, etc).

2 Sample Short Questions

2.1

Given an `int` array called `unluckyNums` that contains at least one element, write an expression that evaluates to `true` if the last element of `unluckyNums` is exactly divisible by 13, and `false` otherwise.

2.2

Write a single statement that declares an int variable that represent the number of columns on a standard TicTacToe board, and gives it an appropriate value. The declaration should be in such a way that nothing in the program can change it after this initialization.

2.3

Rewrite the code below using a just single return statement:

```
if (a && b || !c) {
  return false;
} else {
 return true;
}
```

[*]Note that these are just sample questions from various (and NOT all) topics in the course. The exam may have different form/type of questions. We will provide all necessary java API for the exam (similar to what we did for the midterm).

2.4

```
  public void whatisthis() {                           |  // Show variable values here.
    String s = "Happy holidays to all!";               |
    int num = s.indexOf(" ", 6);                       |
    String s2 = s.substring(0, num);                   |
    int num2 = s.indexOf(s2);                          |
    int i = 0;                                         |
    while(i < s.length()) {                            |
      int x = s.indexOf("o", num2);                    |
      if (x >= 0) {                                    |
        System.out.println(s.substring(num2, x));      |
        num2 = x + 1;                                  |
        i = num2;                                      |
      } else {                                         |
        System.out.println("all");                     |
        i = s.length();                                |
      }                                                |
      i++;                                             |
    }                                                  |
  }                                                    |
```

What is printed by the method `whatisthis`?

Also, show (above) what values the variables have as the code executes (just for the first iteration of while loop).

3 Class Completion and OO

3.1

Complete the following code for class `LibraryBook`:

```
 /** A book in a library with
a title, an author, and a status indicating
  * whether or not it has been loaned out. */
public class LibraryBook {
  // Declare any necessary instance/static variables here.



  /** A new book record with title t, author a, and loan status l. */
  public LibraryBook ( String t, String a, boolean l ) {



  }

  /** Get the title of this book. */
  public String getTitle() {
  }

  /** Get the author of this book. */
  public String getAuthor() {




  }

  /** Get whether or not this book is currently loaned. */
  public boolean isLoaned() {




  }

  /** Loan a copy of this book, if it is currently unloaned.  Return
    * true if the book is succesfully loaned, and return false otherwise. */
  public boolean loanBook() {






  }

  /** Return true if b represents the same book as this book, and false
    * otherwise.  Books are considered the same if they have the same title
    * and author. */
  public boolean equals( LibraryBook b ) {








  }
}
```

3.2

Based on the `LibraryBook` class, complete the following code. Both design and correctness are important. Keep in mind that some of the methods can and should be called by other methods as helper methods.

```
 /** A library
with a catalogue of books available. */
   public class Library {
   // Declare any necessary instance/static variables here





   /** A new library that can hold a maximum of n books */
   public Library( int n ) {






   }

   /** Add a book to the library's catalogue. If the library is full
     * or if a book with that title is already in the library,
     * then it should not be added. Return true if the book is added, and
     * false otherwise. */
   public boolean addBook( LibraryBook b ) {
















   }

   /** Loan a copy of book b.  A book cannot be loaned out if it is not
     * in the library's catalogue or if it is currently loaned out.
     * Return true if the book is successfully loaned, and false otherwise. */
   public boolean loanBook( LibraryBook b ) {
   }


   /** Remove the book with title t and author a from the library.
     * This can only be done if the book is in the library's catalogue
     * and it is not currently loaned.  Return true if the book is removed,
     * and false otherwise. */
   public boolean removeBook( String t, String a ) {



   }


   /** Transfer book b to library l.  Book b can only be transferred from
     * this library if b is in this library's catalogue and it is not currently
     * loaned, and if there is enough room in library l.  Return true if the book
     * is transferred, and false otherwise.  You may assume that b is not
     * already in l. */
   public boolean transfer(LibraryBook b, Library l) {



   }
}
```

4 Strings

Complete the following method. You may NOT use arrays but you are encouraged to use `StringTokenizer`.

```
/**
 * Return the name and surname of the student with maximum grade in studentList.
 * If there are more than one students with highest grade, return
 * the first one.
 *
 * studentList is a comma-separated string of student information in the form:
 * studentName:studentSurname:studentGrade
 *
 * Here is an example:
 * "Mike:Cameron:57.8,Henry:Harris:87.8,Mary:Harts:73,Lisa:Fikes:87.8"
 * For the above example, the method returns "Henry Harris"
 * You may assume that there will be at least one student in studentList. */
 public double getBestStudent(String studentList) {









  }
```

5 Arrays

Complete the following methods (assume all methods are in the same class). You will be marked on correctness, and design.

```
/** Sum the contents of each row of table, and return the sum of
 *  each row from table  in a 1D array of integers.
 *  The sum of row 0 should be in element 0 of the
 *  result, the sum of row 1 in element 1 of the result, and so on.
 *  Assume table is not null, and each of its rows are not null either.*/
```

```
  public static int[] sumRows(int[][] table ) {









  }


/** Return true if all elements of list have the same value, and
  * return false otherwise.
  * Assume list is not null */
  public static boolean allAreEqual( int[] list ) {









  }


 /** Return true if all rows of table have the same sum, and return
  *  false  otherwise.
  *  Assume table is not null, and each of its rows are not null either.
  *  Hint: use sumRows and allAreEqual  methods.*/

  public static boolean sameSumRows (int[][] table ) {









  }
```

6 Sorting/Search

Write a `public static` method called `twoSort` that takes an `int[]` and partially sorts its elements as follows: the elements with even indices are sorted relative to each other and the elements with odd indices are sorted relative to each other. For instance, if the original list is:

```
  0   1   2   3   4   5   6
 ---------------------------
| 5 | 4 | 3 | 2 | 6 | 0 | 1 |
 ---------------------------
```

then the following is partially sorted:

```
  0   1   2   3   4   5   6
 ---------------------------
| 1 | 0 | 3 | 2 | 5 | 4 | 6 |
 ---------------------------
```

The `ints` in the array may occur more than once. Design, and correctness are both important. You should avoid creating/copying unnecessary arrays You may add helper methods if necessary. Also, you may use any sort method or even invent your own.

```
public static void twoSort(int[] list) {




}
```

7 Testing

Write proper test suites for `twoSort` method from previous question. It is enough jut to mention scenarios like bellow.

- list of 1 element: `{4}`, expected result: `{4}`
- list of 2 elements: `{5,1}`, expected result: `{5,1}`
- etc

Similarly, write proper test suites for `sumRows` method from the question before previous question.

8 File I/O, String, Loop

Complete the following methods. You may assume that all necessary import statements have been made.

```
/** Given a filename f and a string s, return true if s is a
 *  line of the file, and return false otherwise.
 *  No need to check for errors: assume s is not null and file f exits.
 *
 *  @param f The name of the file to look in.
 *  @param s The line to look for in the file. */
  public boolean matchesLine(String f, String s) throws IOException {















  }

 /** Write the contents of string s to the new file ‘‘repeat.txt’’ n times
 *   each in a new line.
 *   No need to check for any errors: assume s is not null.
 *
 *  @param s The text to write to the file.
 *  @param n The number of times to write the text to the file. */
 public boolean writeRepeat(String s, int n) throws IOException {















 }
```