Student Name (Last name, First name): ______________________ Student ID: ______________________

Department of Computing

COMP2021 Object-Oriented Programming Quiz

Solutions

Semester 1, 2023/2024

Section A. Multiple Choice Questions (4 points)

For each question, only one answer from the listed choices is correct. Write the corresponding letter in the parentheses.

Question 1. Read the following program:

```
public class HelloWorld{
    public static void main(String [] args) {
        int x = 3;
        int y = 1;
        if (x = y)
            System.out.println("Not equal");
        else
            System.out.println("Equal");
    }
}
```

What is the result if we try to compile and execute the program? (C)

A. The program compiles successfully and outputs “Equal” when executed.

B. The program compiles successfully and outputs “Not equal” when executed.

C. The program does not compile successfully.

D. The program compiles successfully and produces no output when executed.

Question 2. Which of the following statements is true? (A)

A. Methods cannot be overridden as private.

B. static methods cannot be overloaded.

C. private methods cannot be overloaded.

D. In an overloaded method of a subclass, methods of the corresponding superclass cannot be called directly.

Section B. True or False (12 points)

Statements in Questions 3 through 5 concern object-oriented programming in Java. Please indicate in the table below whether each statement is correct. Put a T in an empty table cell if the corresponding statement is correct; Otherwise, put an F. A correct T or F is worth 1 point, and an incorrect T or F or an empty cell is worth 0 points.

Question 3				Question 4				Question 5			
a	b	c	d	a	b	c	d	a	b	c	d
F	F	T	F	F	F	T	F	F	T	T	F

Question 3. Java basics

a. Given int x = 9, y = 4;, the result of x / y is 2.25.

b. Given an 8-bit signed value x = $(10000000)_2$ in two's complement, the value of x is -127 in decimal.

c. A float value takes 4 bytes in java.

d. Given int x = 2;, if(x < 2){ x++; } elseif (x > 2) { x--; } is a valid statement.

Question 4. Object-based programming

a. Given String s = "Hello.World ";, s.charAt(7) returns 'W'.

b. Keyword final cannot be used on a formal parameter. For example, class X { void f(final int x){ } } will cause a compilation error.

c. Methods do not take any space in an object in java.

d. Given two packages a.b and c.d, importing both a.b.* and c.d.* will cause a compilation error if each of them contains a class named E.

Question 5. Object-oriented programming

a. Given definition class Hero { public boolean equals(Hero h){ return false; } }, class Hero overrides the equals method inherited from Object to always return false.

b. Given two classes A and B, with B inheriting from A. If A has a private instance field int f, every object of B has an instance field f of type int.

c. All classes in Java directly or indirectly inherit the Object class.

d. private methods defined in the parent class cannot be redefined.

Section C. Short-Answer Questions (16 points)

Question 6. (8 points) See below a method header from the **StringBuffer** class:

public StringBuffer append(**String** s)

1) What is the role of each word below in defining the method? (6 points)

public: The method is accessible everything in the program.

StringBuffer: The method returns a reference to a StringBuffer object.

String: The method expects as the input a reference of String type.

2) If we include one more keyword **static** to the header, how should we call this method with an input parameter named str of type **String**? Please write the code. (2 points)

StringBuffer.append(str);

Or

StringBuffer strBuf = …;

strBuf.append(str);

Question 7. (8 points) A Java class that does not implement information hiding faces two significant risks. First, a piece of client code may misuse and corrupt its objects. Second, a piece of client code may depend unnecessarily on its internal implementation details. Please

1) use example code snippets to explain the two risks (4 points), and

2) describe how information hiding could help avoid both risks in Java (4 points).

See example code snippets and corresponding explanations on Slides 4 through 8 of LEC04 Object-Based Programming (II).

Section D. Programming (18 points)

Question 8. (8 Points) Method toPigLatinWord in Listing 1 translates and English word into Pig Latin. Each English word contains only English letters is translated into a Pig Latin word based on the following rules:

1. All the leading non-vowel letters are appended with the letters “ay” and moved to the end of the word, in their original order. Vowel letters include ‘a’, ‘e’, ‘i’, ‘o’, ‘u’, ‘A’, ‘E’, ‘I’, ‘O’, and ‘U’.

2. The first letter of the resultant word should be in uppercase if and only if the first letter of the original word is in uppercase. All the other letters in the resultant word should be in lowercase.

For example, English words “The”, “FinTech”, “hmm”, and “out” will be translated into “Ethay”, “Intechfay”, “hmmay”, and “outay”, respectively, in Pig Latin.

Notes:

- Your implementation will be graded solely based on code correctness rather than efficiency.

- Your implementation can only invoke the following methods from classes String and Character:

```
// From class String
public int charAt(int index);   // Returns the char value at the specified index.
public int indexOf(char ch, int fromIndex); // Returns the index within this string of the
    // first occurrence of the specified character, starting the search at the specified
    // index. If no such character occurs in this string at or after position fromIndex,
    // then -1 is returned.
public int length();     // Returns the length of this string.
public String subString(int beginIndex, int endIndex); // Returns a string that is a
    // substring of this string. The substring begins at the specified beginIndex and
    // extends to the character at index endIndex - 1.
public String toLowerCase(); // Converts all characters in this String to lowercase.

// From class Character
public static boolean isUpperCase(char ch); // Determines if the character is in uppercase.
public static char toUpperCase(char ch); // Converts the character argument to uppercase.
```

```
                          Listing 1. Class PigLatin

public class PigLatin{
    private static String toPigLatinWord(String word){
        // Translates ‘word’ into Pig Latin.
        // Each word is not null and contains a non-empty sequence of English letters
        // Complete this method
        String vowels = "aoeiuAOEIU";
        boolean isCapitalized = Character.isUpperCase(word.charAt(0));
        String wordInLower = word.toLowerCase();

        // get leading non-vowels
        int start = 0;
        while(start<wordInLower.length() && vowels.indexOf(wordInLower.charAt(start)) < 0)
            start++;
        String prefix = wordInLower.substring(0, start);
        String postfix = start >= wordInLower.length() ? "" : wordInLower.substring(start);

        String newWord = postfix + prefix + "ay";
        if(isCapitalized)
            newWord = Character.toUpperCase(newWord.charAt(0)) + newWord.substring(1);
        return newWord;
    }
}
```

Question 9. (10 Points) In the Western calendar, a year is divided into 12 months, numbered from 1 to 12, and months into days, numbered starting from 1. The 1st, 3rd, 5th, 7th, 8th, 10th, and 12th months have 31 days each, while the 4th, 6th, 9th, and 11th months have 30 days each. The 2nd month usually has 28 days, but it has 29 days in leap years. A year is a leap year if and only if either it can be divided by 4 but not by 100, or it can be divided by 400. For example, 2000 and 2004 are leap years, but 2011 and 2100 are not.

Class MyDate in Listing 2 abstracts dates in the Western calendar. Please read the class and the comments in its methods carefully, then complete the methods as required so that:

- The JUnit test method MyDateTest.proceedTest shown in Listing 3 can execute successfully.

- The program should terminate immediately (by invoking System.exit(1);) whenever the parameter(s) used to invoke a method is/are invalid.

```
                          Listing 2. Class MyDate

public class MyDate {
    private int day, month, year;
    public MyDate(int day, int month, int year) {
        setDate(day, month, year);
    }

    public int getDay() { return day; }
    public int getMonth() { return month; }
    public int getYear() { return year; }
    public static boolean isValidYear(int year){ return year > 0; }
    public static boolean isValidMonth(int month){ return 1 <= month && month <= 12; }
    public static boolean isValidDate(int day, int month, int year){
        // Checks if 'day', 'month', and 'year' denote a valid date.
        int numDays = getNumberofDays(month, year);
        return 1 <= day && day <= numDays;
    }
    public void setDate(int day, int month, int year) {
        // Uses 'day', month', and 'year' to set this date if they denote a valid date.
        if(!isValidDate(day, month, year)) System.exit(1);
        this.day = day;
        this.month = month;
        this.year = year;
    }
    public static boolean isLeapYear(int year){
        // Check if 'year' is a leap year.
        if(!isValidYear(year)) System.exit(1);
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
    }
    public static int getNumberofDays(int m, int y){
        // Compute the number of days in month 'm' of year 'y'.
        if(!isValidMonth(m) || !isValidYear(y)) System.exit(1);
        switch(m){
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                return 31;
            case 4: case 6: case 9: case 11:
                return 30;
            default:
                return isLeapYear(y) ? 29 : 28;
        }
    }
    public void proceed(){
        // Proceeds to the next date.
        // For example, if ‘this’ date is Dec. 31, 2023 before the call,
        //              it represents Jan. 1, 2024 afterward.
        // Complete this method
        int numDaysInMonth = getNumberofDays(month, year);
        if(day < numDaysInMonth){
            day++;
        }
        else{ //  day == numDaysInMonth
            day = 1;
            if(month < 12){
                month++;
            }
            else{
                month = 1;
                year++;
            }
        }
    }
    public String toString(){ return day + "/" + month + "/" + year; }
}
```

```
                              Listing 3. Test class MyDateTest

import org.junit.Test;
import static org.junit.Assert.*;

public class MyDateTest{
    @Test
    void proceedTest() {
        MyDate date1 = new MyDate(25, 11, 2023);
        for(int i = 0; i < 10; i++){ date1.proceed(); }
        assertEquals("5/12/2023", date1.toString());

        MyDate date2 = new MyDate(20, 2, 2023);
        for(int i = 0; i < 10; i++){ date2.proceed(); }
        assertEquals("2/3/2023", date2.toString());
    }
}
```

-- End --