Name: ____

SMU
SINGAPORE MANAGEMENT UNIVERSITY

AY 2021-22 Term 2

CS102 Programming Fundamentals II

QUESTION BOOKLET
INSTRUCTIONS TO CANDIDATES

(Please read this page only. Do not turn over this page until instructed.)

1. This is a **close-book** examination. Please remove everything from your workspace except for laptop, pens, pencils, erasers, tissues and staplers before you begin.

2. You are to place all other items (including notes, documents, calculator, mobile and communication devices, SMART watches, Fitness Trackers, wallet and all other personal belongings) in your bag. You must place your bags and personal belongings at the front of the examination hall. Mobile devices are to be **SWITCHED OFF.**

3. The time allowed for this examination paper is **2 hours**.

4. This paper contains a total of **10** MCQs and **4** open questions and comprises **24** pages.

5. This examination consists of 2 parts.
   a. **Multiple choice** questions are worth 2 marks each. For each question, select exactly **ONE** choice.

   b. **Open** questions are worth the marks as stated in the questions. Please follow the instructions for each of these questions.

6. You are required to input all answers (both sections **A** and **B)** on eLearn Quiz (202122T2-Final).

7. You must **return** all parts of this exam paper to the instructors. Missing exam papers or parts thereof will be considered cheating.

8. There shall be NO ENQUIRY during the exam. I
   a. For MCQ, If there is any ambiguity, pick the best answer.
   b. For Q11 - 14, state any assumptions if necessary when you answer the questions.

	Marks
MCQ	20
Q11	9
Q12	8
Q13	7
Q14	6
TOTAL	50

Section A: MCQ (20 marks)

1. **[Difficulty: **]** Consider the following Java class:

```
class MysteryException extends Exception {

}

public class Q1 {
    public static char doMagic(String s, int a)
            throws MysteryException {
        //here
    }

    public static void main(String[] args) throws MysteryException {
        System.out.println(doMagic(null, 0));
    }
}
```

Which of the following code is valid (no compile error) when inserted at "`//here`"?

I.
```
return s.charAt(a);
```

II.
```
throw new MysteryException();
```

III.
```
return (char) a;
```

IV.
```
try {
    return 'B';
} catch (Exception e) {
    return 'A';
}
```

A. I

B. II, IV

C. I, II, III

D. II

E. All the above

2. **[Difficulty: *]** What happens when you try to compile and execute the following class?

```
interface Presentable {
}

class Student implements Presentable {
}

public class Q2 {
    public static void main(String[] args) {
        Student obj = null;
        System.out.println(obj instanceof Presentable);
    }
}
```

A. `Q2` cannot compile because the `instanceof` keyword can only be used as a condition within an if-statement.

B. `Q2` cannot compile because `instanceof` cannot be used on an interface, i.e. `Presentable`.

C. Runtime error.

D. `Q2` compiles and runs to print `true`.

E. `Q2` compiles and runs to print `false`.

3. **[Difficulty: *]** Consider the following code:

```java
import java.util.*;

class Person {
   private int age;

   public Person(int age) {
       this.age = age;
   }

   public String toString() {
       return String.valueOf(age);
   }
}

public class Q3 {
   public static void main(String[] args) {
       Map<String, Person> map = new HashMap<>();

       Person p = new Person(2);
       map.put("amy", p);
       map.put("bill", p);

       p = new Person(3);
       map.put("amy", p);
       map.put(null, new Person(4));
       System.out.println(map.get("amy"));
       System.out.println(map.get("bill"));
       System.out.println(map.get(new Person(3)));
       System.out.println(map.get(null));
   }
}
```

What is the output when `Mystery` is compiled and executed?

A.
```
3
2
null
4
```

B.
```
3
3
null
4
```

C.
```
3
3
null
null
```

D. Compile error.

E. Runtime error.

4. **[Difficulty: *]** Given the following classes:

```
class Fruit {
    public String name;

    public Fruit(String name) {
        this.name = name;
    }

    public String toString() {
        return name;
    }

    public Fruit clone() {
        return new Fruit(name);
    }
}

public class Q4 {
    public static void main(String[] args) {
        Fruit f1 = new Fruit("A");
        Fruit f2 = new Fruit("B");
        Fruit f3 = f2;

        Fruit[] all = { f2, f1, f3 }; // note the order
        Fruit[] copy = all;
        f1 = f2;

        copy[0] = new Fruit("C");
        copy[1] = f1.clone();
  }
}
```

At the end of the execution of this code, how many `Fruit` objects are created, and how many are eligible for garbage collection?

Garbage collection implies that objects that are no longer referenced by the program are "garbage" and can be thrown away by Java.

	Number of `Fruit` objects created	**Number of `Fruit` objects for garbage collection**
A.	4	0
B.	4	1
C.	3	0
D.	3	1
E.	3	2

5. **[Difficulty: **]** Consider the following classes:

```
package first;

public class Product {
    protected String doSomething() {
        return "A";
    }
}
```

```
package second;

import first.*;

public class Pen extends Product {
}
```

```
1 package second;
2
3 public class Test {
4     public static void main(String[] args) {
5         Pen p = new Pen();
6         System.out.println(p.doSomething());
7     }
8 }
```

Which of the following is **TRUE**?

A. There is a compilation error in `Product.java`. The following declaration is missing in the `Product` class:

```
public Product() { }
```

B. There is a compilation error in `Pen.java`. The following declaration is missing in the `Pen` class:

```
public Pen() {
    super();
}
```

C. There is a compilation error in `Test.java`. You need to add the following import statement

```
import first.*;
```

D. There is a compilation error at Line 6 of `Test` class. The `Pen` class has no accessible `doSomething` method.

E. There is no compilation error and it will print `A` to the console.

6. **[Difficulty: **]** What is the output when the `Test` class is compiled and executed?

```
interface Human {
    public String talk();
}
class Student implements Human {
    public String talk() { return "S"; }
}
class TA extends Student {
    public String talk() { return "T"; }
}

public class Test {
    public static void main(String[] args) {
        Human[] humanList = {new Student(), new TA()};
        Student[] sList = {new Student(), new TA()};

        for (Student s : sList) {
            System.out.print(s.talk()  + " ");
        }
        System.out.println();

        for (Human h : humanList) {
            if (h instanceof Human) {
                System.out.print(h.talk());
            }
            if (h instanceof Student) {
                System.out.print(h.talk());
            }
            if (h instanceof TA) {
                System.out.print(h.talk());
            }
            System.out.print(" ");
        }
        System.out.println();
    }
}
```

A.
```
S T
S T
```

B.
```
S S
SS STT
```

C.
```
S S
SS SST
```

D.
```
S T
SS SST
```

E.
```
S T
SS TTT
```

7. **[Difficulty: **]** Consider the following Java classes:

```
import java.util.Arrays;

class Person {
    private int age;

    public Person(int age) {
        this.age = age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String toString() {
        return String.valueOf(age);
    }
}

public class Test {
    public static void doMagic(Person[] arr) {
        arr[0].setAge(4);
        arr = new Person[]{new Person(9), new Person(8), new Person(7)};
        arr[1] = new Person(5);
    }

    public static void main(String[] args) {
        Person[] arr = {new Person(1), new Person(2), new Person(3)};

        doMagic(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

What is the output when the `Test` class is compiled and executed?

A. `[1, 2, 3]`

B. `[4, 2, 3]`

C. `[1, 5, 3]`

D. `[9, 5, 7]`

E. `[9, 8, 7]`

8. **[Difficulty: *]** Given the following code:

```
import java.util.*;

public class Q8 {
    public static void main(String[] arg) {
        //here
        a.add("apple");
        a.add("berry");
        a.add("cherry");
        a.add("dragonfruit");
        a.add("apple");
        System.out.println(a);
    }
}
```

What can be inserted at "`//here`" to produce the following output?

```
c:\>java Mystery
[apple, cherry, berry, dragonfruit]
```

I. `List<String> a = new ArrayList<>();`

II. `HashSet<String> a = new HashSet<>();`

III. `Set<String> a = new Set<String>();`

IV. `Set<Object> a = new TreeSet<>();`

V. `HashSet<String> a = new TreeSet<>();`

A. II

B. IV

C. II & III

D. III & IV

E. I & III

9. **[Difficulty: *]** What is the output when you compile and run the following program?

```
public class Q9 {
    private String s;
    private int b;
    private double d;

    public String toString() {
        return b + d + s + b + d;
    }

    public static void main(String[] args) {
        System.out.println(new Q9());
    }
}
```

A. `0.0null0.0`

B. `0.00.0`

C. `00.0null00.0`

D. `0.0null00.0`

E. `00.000.0`

10. **[Difficulty: *]** Which of the following code segments when inserted at "`//here`" will not cause a compile error?

```
import java.util.ArrayList;

class Person {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

public class Q10 {
    public static void main(String[] args) {
        final Person p = new Person();
        //here
    }
}
```

I. `Person q = p;`

II.
```
Person[] arr = {p};
arr[0] = new Person();
```

III. `p.setName("John");`

IV. `p = new Person();`

A. I & II

B. I & III

C. II & III

D. I, II & III

E. All the above.

Section B

Question 11 [Difficulty:*, 9 marks]

implement the method called `isUnique`. This method takes in 1 parameter:

a. `array` (type: `int[]`): An array of integer values. Assume non-`null`.

This method returns `true` if there is no duplicate value in the `array`, i.e. each value in the array only appears once. Otherwise, it returns `false`.

The output for the following program is:

```
isUnique([3, 1, 2]): true
isUnique([3, 1, -2, 2, -1, -3]): true
isUnique([3, 1, 2, -1, -2, -3, 0, -3, -3]): false
isUnique([]): true
isUnique([1]): true
```

```
import java.util.*;

public class Q11Test {
   public static void main(String[] args) {
        {
            int[] arr = {3, 1, 2};
            System.out.printf("isUnique(%s): %b%n",
                    Arrays.toString(arr), Q11.isUnique(arr));
        }
        {
            int[] arr = {3, 1, -2, 2, -1, -3};
            System.out.printf("isUnique(%s): %b%n",
                    Arrays.toString(arr), Q11.isUnique(arr));
        }
        {
            // -3 appeared 3 times
            int[] arr = {3, 1, 2, -1, -2, -3, 0, -3, -3};
            System.out.printf("isUnique(%s): %b%n",
                    Arrays.toString(arr), Q11.isUnique(arr));
        }
        {
            int[] arr = {};
            System.out.printf("isUnique(%s): %b%n",
                    Arrays.toString(arr), Q11.isUnique(arr));
        }
        {
            int[] arr = {1};
            System.out.printf("isUnique(%s): %b%n",
                    Arrays.toString(arr), Q11.isUnique(arr));
        }
    }
}
```

```
// Answer:

public class Q11 {
   public static boolean isUnique(int[] array) {
        // your answer
    }
}
```

Question 12 [Difficulty:, 8 marks]**
Given the class diagram in the appendix 1.

A. [5 marks] Implement `MobileSubscription` class according to the class diagram.

a. Every `MobileSubscription` object has a fixed number of `talkTime`.

b. `MobileSubscription` A is considered the same as `MobileSubscription` B if and only if
   i. The `equals` method of `Subscription` returns `true` AND
   ii. the number of `talkTime` is the same.

c. `toString` method will returns a String object whose value is of the following format:

```
{name='<name>', startDate='<startDate>', talkTime=<talkTime>}
```

Example:

```
{name='EX Core', startDate='2022-05-04', talkTime=500}
```

d. `compareTo` method is used to sort `MobileSubscription` objects based on the following orders:

- `startDate` in ascending order
- `name` in ascending order
- `talkTime` in ascending order

```
// Answer:
```

B. **[3 marks]** Complete the `getMobileSubscriptionsBetween` in `Utility`.
This method returns the `MobileSubscription` objects that start between `startDate`(inclusive) and `endDate` (exclusive). The result should be sorted by their date in ascending order (i.e. chronological order, oldest date first).

Note:
1. The objects passed into this method may contain other `Subscription` objects or its subclasses.
2. The method `now()` in the `SMUDate` class returns today's date.
3. Assume `subscriptions` parameter is non-`null`.

The output of `UtilityTest` is shown below:

```
2022-05-01
2022-05-02
2022-05-04
2022-05-04
2022-05-04
2022-05-05
```

The source code for `UtilityTest` class is given on the next page:

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class UtilityTest {
   public static void main(String[] args) {
       List<Subscription> values = List.of(
               new MobileSubscription("apple", new SMUDate(1, 5, 2022),500),
               new MobileSubscription("orange", new SMUDate(29, 4, 2022),500),
               new MobileSubscription("pear", new SMUDate(2, 5, 2022),500),
               new MobileSubscription("papaya", new SMUDate(4, 5, 2022),500),
               new MobileSubscription("papaya", new SMUDate(4, 5, 2022),500),
               new Subscription("papaya", new SMUDate(4, 5, 2022)),
               new MobileSubscription("papaya", new SMUDate(4, 5, 2022),500),
               new NewspaperSubscription("papaya", new SMUDate(4, 5, 2022),"SPH"),
               new MobileSubscription("papaya", new SMUDate(5, 5, 2022),500)
               );
       List<MobileSubscription> result = Utility.getMobileSubscriptionsBetween(
               values,
               new SMUDate(1, 5, 2022),
               new SMUDate(9, 5, 2022));
       for (MobileSubscription m : result) {
           System.out.println(m.getStartDate());
       }
   }
}
```

```java
// Answer:

public class Utility {
   public static List<MobileSubscription> getMobileSubscriptionsBetween(
            List<Subscription> subscriptions, SMUDate start, SMUDate end) {
        // your answer
    }
}
```

Question 13 [Difficulty:, 7 marks]**

Implement the method called `guessWordle`. This method takes in 2 parameters:

a. `guess` (type: `String`): The 5-letter word guessed by the user. Assume the value is non-`null`, consists of 5 characters and is in uppercase.
b. `correct` (type: `String`): The correct 5-letter word. Assume the value is non-`null`, consists of 5 characters and is in uppercase.

This method compares each character in `guess` against `correct`.

1. If the character in `guess` is found at the same spot of `correct`, the character `'G'` is included in the return string.
2. If the character in `guess` is found at a different spot of `correct`, the character `'Y'` is included in the return string. If the character is repeated in `guess` and there is only one occurrence of the character in `correct`, then only the 'Y' character is returned for the first occurrence of the character in `guess`. For example, `guessWordle("SHEER", "SLOPE")` returns `"G-Y--"`.
3. Otherwise, the character `'-'` is included in the return string.

If the correct word is `"BINGO"` and the guess is `"BRIDE"`, then `"G-Y--"` is returned.

- `'G'` as the character **'B'** of "**B**RIDE" is in the word and at the right spot of "**B**INGO".
- `'-'` as the character **'R'** of "B**R**IDE" is not found in the word "BINGO".
- `'Y'` as the character **'I'** of "BR**I**DE"(the 3rd character) is in the word but at the wrong spot of "BINGO" (the 2nd character).
- `'-'` as the character **'D'** of "BRI**D**E" is not found in the word "BINGO".
- `'-'` as the character **'E'** of "BRID**E**" is not found in the word "BINGO".

The output of `Q13Test` is as follow:

```
public class Q13Test {
    public static void main(String[] args) {
     {
       String guess = "PETAL";
       String correct = "SCRUM";
       System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
               guess, correct, Q13.guessWordle(guess, correct));
     }
     {
       String guess = "ROYAL";
       String correct = "PUPIL";
       System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
               guess, correct, Q13.guessWordle(guess, correct));
     }
     {
       String guess = "SPEAR";
       String correct = "SMACK";
       System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
               guess, correct, Q13.guessWordle(guess, correct));
     }
     {
       String guess = "SPOOL";
       String correct = "TOOLS";
       System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
               guess, correct, Q13.guessWordle(guess, correct));
     }
```

```
      {
        String guess = "TOOLS";
        String correct = "SWORN";
        System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
                guess, correct, Q13.guessWordle(guess, correct));
      }
      {
        String guess = "SHEER";
        String correct = "SLOPE";
        System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
                guess, correct, Q13.guessWordle(guess, correct));
      }
      {
        String guess = "BRIDE";
        String correct = "BINGO";
        System.out.printf("Guess  :%s%nCorrect:%s%nResult :%s%n%n",
                guess, correct, Q13.guessWordle(guess, correct));
      }
    }
}
```

```
Guess  :PETAL
Correct:SCRUM
Result :-----

Guess  :ROYAL
Correct:PUPIL
Result :----G

Guess  :SPEAR
Correct:SMACK
Result :G--Y-

Guess  :SPOOL
Correct:TOOLS
Result :Y-GYY

Guess  :TOOLS
Correct:SWORN
Result :--G-Y

Guess  :SHEER
Correct:SLOPE
Result :G-Y--

Guess  :BRIDE
Correct:BINGO
Result :G-Y--
```

```
// Answer:
public class Q13 {
    public static String guessWordle(String guess, String correct) {
        // your answer
    }
}
```

Question 14 [6 marks, Difficulty: *]**

`Utility4` is a buggy implementation of your Lab Test Q3 Part 4. Class diagram is given on Appendix B.

There are **<u>more than 1</u>** errors in this implementation. Identify and correct **ALL** execution and logic errors (i.e., errors that cause the program to behave incorrectly when compiled and executed).

```
$java Utility4
Test 1
Result:[IS111, IS112]
Result:[IS111, IS112]

Test 2
Result:[IS111, CS1]
Result:[IS111, CS1]

Test 3
Result:[IS111, IS113]
Result:[IS111, IS113]
```

The description of the question is as follows:

Implement `getCoursesTakenInSameYearAndSemesterOfStudy(Utility4.java)`. It takes in two parameters:

a. `courseList1` (type: `List<Course>`): the list of courses taken by the first student. This attribute is non-null and is not sorted in any order.
b. `courseList2` (type: `List<Course>`): the list of courses taken by the second student. This attribute is non-null and is not sorted in any order.

It returns the code of the courses that both students take in the same year(first, second, third or fourth year of study) and semester of study (semester 1 or 2) sorted in ascending order. See explanation below. You can assume that both students did not take any Leave of Absence (LOA).

For example,

Academic Term	**Student A**	**Student B**
AY202021T1	**IS111** (Year 1 of study, Semester 1)	
AY202021T2	IS112 (Year 1 of study, Semester 2)	**IS111**, IS112 (Year 1 of study, Semester 1)
AY202122T1	**IS113** (Year 2 of study, Semester 1)	IS114 (Year 1 of study, Semester 2)
AY202122T2	IS114 (Year 2 of study, Semester 2)	**IS113** (Year 2 of study, Semester 1)

Both students did
- IS111 in Year 1 of study, Semester 1 (even though at different academic terms AY202021T1 and AY202021T2)
- IS113 in Year 2 of study, Semester 2 (even though at different academic terms AY202122T1 and AY202122T2)

Thus, IS111 and IS113 are included in the result.

Both students did IS112 in the same academic term (AY202021T2) but student A did it in his Year 1 of study, Term 2 whereas student B did it in his Year 1 of study, Term 1. Thus, IS112 is not included in the result.

Note: 1 mark will be deducted for every incorrect error that you have identified. The minimum score for this question is 0 mark.

```
001  import java.util;
002
003  public class Utility4 {
004      /**
005       * This method generates a Map with academic term (e.g. 'AY202021T1')
006       * as the key, and semester of study as the value
007       *   1 for Year 1, term 1,
008       *   2 for Year 1, term 2,
009       *   3 for Year 2, term 1,
010       *   4 for Year 2, term 2,
011       *   ...
012       *
013       * For example,
014       *
015       * System.out.println(generateTerms("AY202021T1", "AT202223T1"));
016       *
017       * displays the following output:
018       *
019       * {AY202021T1=1, AY202021T2=2, AY202122T1=3, AY202122T2=4, AY202223T1=5}
020       *
021       * Note: The order does not matter.
022       */
023      private static Map<String, Integer> generateTerms(
024              String start, String end) {
025
026          int startYear = Integer.parseInt(start.substring(2, 6));
027
028          String startTerm = start.substring(9);
029
030          int endYear = Integer.parseInt(end.substring(2, 6));
031
032          String endTerm = end.substring(9);
033
034          HashMap<String, Integer> result = new HashMap<>();
035
036          if (startYear == endYear) {
037
038              result.put(start, 1);
039
040              result.put(end, 2);
041
042              return result;
043          }
044
045          int termCount = 1;
046
047          if (startTerm.equals("2")) {
048
049              startYear++;
050          }
051
052          if (endTerm.equals("1")) {
053
054              endYear--;
055
056          }
057
058          for (int i = startYear; i <= endYear; i++) {
059
060              String nextYear = ("" + i + 1).substring(2);
061
062              String term1 = "AY" + i + nextYear + "T1";
063
```

```java
            String term2 = "AY" + i + nextYear + "T2";

            result.put(term1, termCount++);

            result.put(term2, termCount++);
        }

        if (endTerm.equals("1")) {

            result.put(end, termCount);

        }

        return result;
    }

    /**
     * Given a list of courses, this method returns a List of the unique academic
     * terms in chronological order.
     *
     * For example,
     *
     * System.out.println(getTerms(List.of(
     *   new Course("IS110", "AY202021T1"),
     *   new Course("IS111", "AY202021T1"),
     *   new Course("IS112", "AY202122T1"),
     *   new Course("IS112", "AY202021T2"))));
     *
     * generates the following output:
     *   {AY202021T1, AY202021T2, AY202122T1}
     */
    private static List<String> getTerms(List<Course> courseList) {

        List<String> terms = new ArrayList<>();

        for (Course c : courseList) {

            String aTerm = c.getTerm();

            if (!terms.contains(aTerm)) {

                terms.add(aTerm);

            }
        }

        return terms;
    }

    /**
     * This method returns a map where the key is the semester of study
     * and the value is the List of Courses taken in that semester.
     *
     * For example,
     *
     * System.out.println(getTermCourseMapping(List.of(
     *   new Course("IS110", "AY202021T1"),
     *   new Course("IS111", "AY202021T1"),
     *   new Course("IS112", "AY202122T1"),
     *   new Course("IS112", "AY202021T2"))));
     *
     * generates the following output:
     *   {1=[IS110(AY202021T1), IS111(AY202021T1)],
     *    2=[IS112(AY202021T2)], 3=[IS112(AY202122T1)]}
     *
     * Note: The order does not matter.
     */
    private static HashMap<Integer, List<Course>> getTermCourseMapping(
            List<Course> courseList) {

        Map<Integer, List<Course>> result = new HashMap<>();

        List<String> terms = getTerms(courseList);

        String firstTerm = terms.get(0);

        String lastTerm = terms.get(terms.size());

        Map<String, Integer> termMapping = generateTerms(firstTerm, lastTerm);

        for (Course c : courseList) {
            String aTerm = c.getTerm();

            int key = termMapping.get(aTerm);

            List<Course> termCourses = result.get(key);

            if (termCourses == null) {

                termCourses = new ArrayList<>();

                result.put(key, termCourses);
            }

            termCourses.add(c);
        }

        return result;
    }

    public static List<String> getCoursesTakenInSameYearAndSemesterOfStudy(
            List<Course> courseList1, List<Course> courseList2) {

        Map<Integer, List<Course>> termCourseMap1 =
               getTermCourseMapping(courseList1);

        Map<Integer, List<Course>> termCourseMap2 =
               getTermCourseMapping(courseList2);


        List<String> result = new ArrayList<>();

        int numTerms = termCourseMap1.size();

        for (int term = 1; term <= numTerms ; term++) {

            List<Course> termCourses2 = termCourseMap2.get(term);

            List<Course> termCourses1 = termCourseMap1.get(term);

            for (Course c1 : termCourses1) {

                for (Course c2 : termCourses2) {

                    if (c1.getCode().equals(c2.getCode())) {

                        result.add(c1.getCode());

                    }
                }
            }
        }

        Collections.sort(result);

        return result;
    }

    public static void main(String[] args) {
    {
           System.out.println("Test 1");
           List<Course> courseList1 = List.of(
                 new Course("IS111", "AY202021T1"),
                 new Course("IS112", "AY202021T2"));
           List<Course> courseList2 = List.of(
                 new Course("IS111", "AY202122T1"),
                 new Course("IS112", "AY202122T2"));
           List<String> actual = getCoursesTakenInSameYearAndSemesterOfStudy(
                    courseList1, courseList2);
           System.out.println("Result:[IS111, IS112]");
           Collections.sort(actual);
           System.out.println("Result:" + actual);
           System.out.println();
       }

       {
       System.out.println("Test 2");
       List<Course> courseList1 = List.of(
            new ExchangeCourse("CS1", "AY202122T1", new University("apple", "A")),
            new Course("IS114", "AY202021T2"),
            new Course("IS113", "AY202122T2"),
            new Course("IS111", "AY202021T1"));
       List<Course> courseList2 = List.of(
            new Course("IS111", "AY202122T2"),
            new ExchangeCourse("CS1", "AY202223T2", new University("apple", "A")),
            new Course("IS112", "AY202223T1"));
       List<String> actual = getCoursesTakenInSameYearAndSemesterOfStudy(
            courseList1, courseList2);
           System.out.println("Result:[CS1, IS111]");
           System.out.println("Result:" + actual);
           System.out.println();
       }
       {
       System.out.println("Test 3");
       List<Course> courseList1 = List.of(
           new Course("IS111", "AY202021T1"),
           new Course("IS113", "AY202122T1"),
           new Course("IS114", "AY202122T2"),
           new Course("IS112", "AY202021T2"));
       List<Course> courseList2 = List.of(
           new Course("IS114", "AY202122T1"),
           new Course("IS111", "AY202021T2"),
           new Course("IS112", "AY202021T2"),
           new Course("IS113", "AY202122T2"));
       List<String> actual = getCoursesTakenInSameYearAndSemesterOfStudy(
           courseList1, courseList2);
       System.out.println("Result:[IS111, IS113]");
       System.out.println("Result:" + actual);
       System.out.println();
       }
   }
}




```

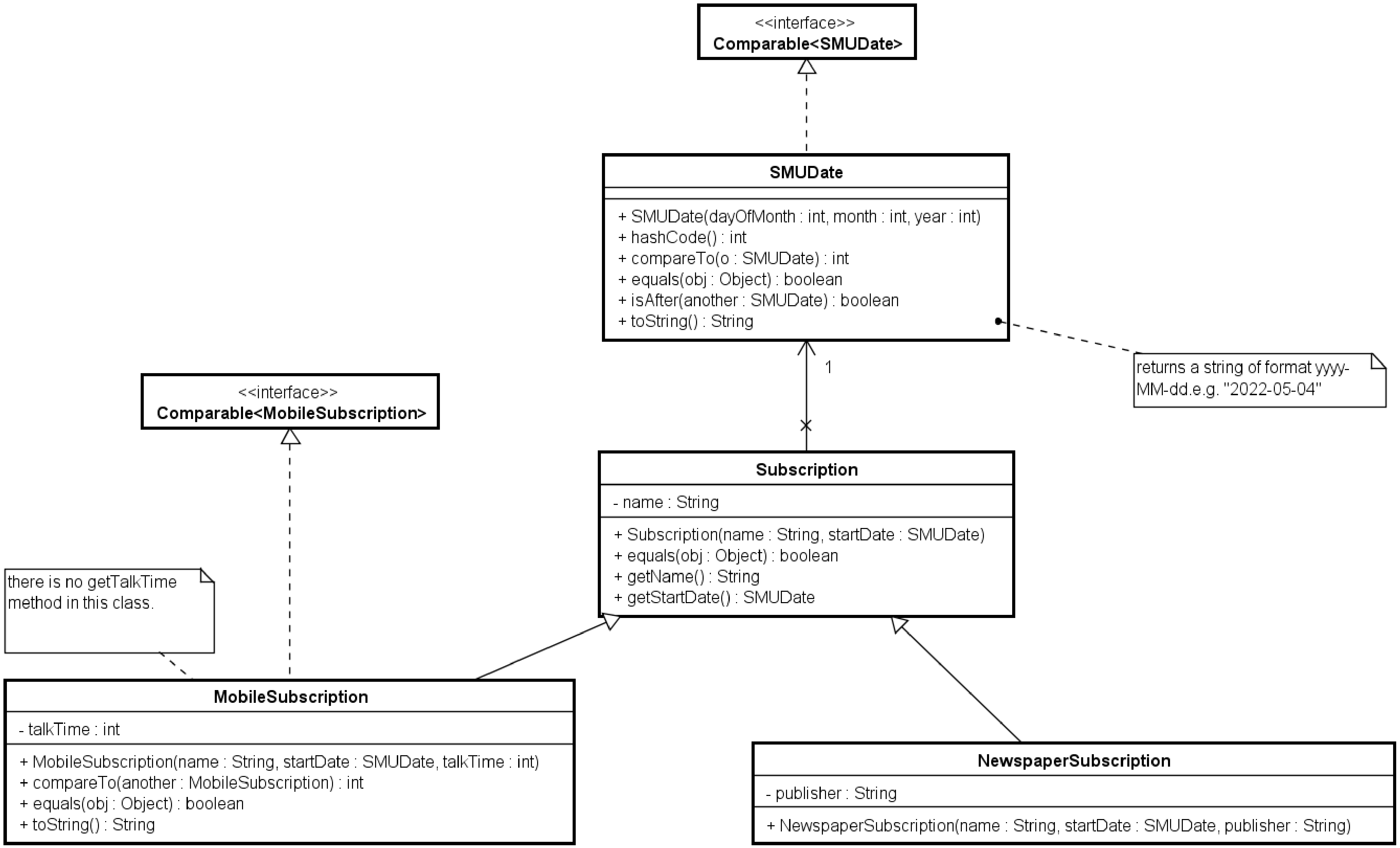
For each error you have identified, please clearly state the following three pieces of information in one line, separated by colons.

```
<Line number>:<addition|modify|deletion>:<corrected statement for add & modify>
```

For example, we have identified "mistake 0" (exclude this mistake in your submission) at line 00]. It should be described as:

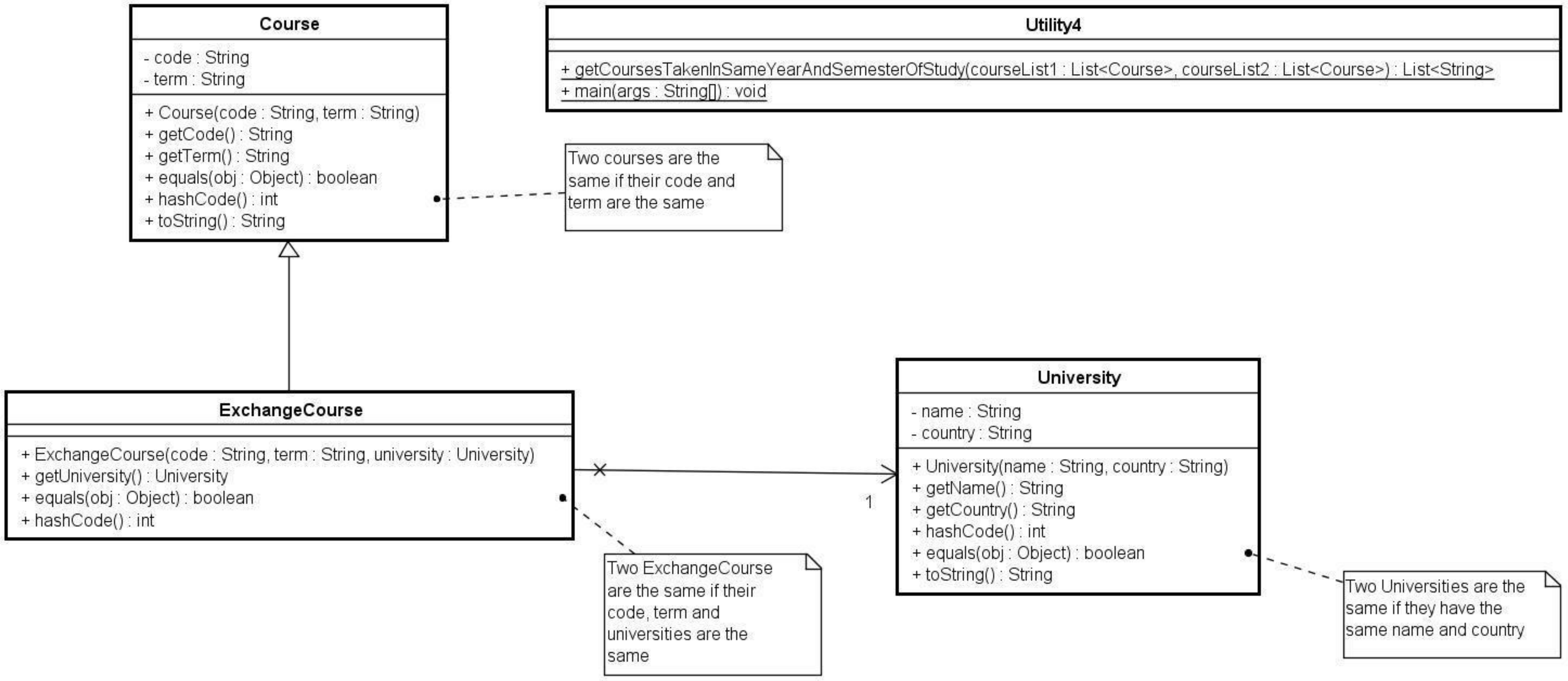
```
001:mod:import java.util.*;
```

APPENDIX A



<<interface>>
Comparable<SMUDate>
SMUDate
+ SMUDate(dayOfMonth : int, month : int, year : int)
+ hashCode() : int
+ compareTo(o : SMUDate) : int
+ equals(obj : Object) : boolean
+ isAfter(another : SMUDate) : boolean
+ toString() : String
returns a string of format yyyy-MM-dd.e.g. "2022-05-04"
1
<<interface>>
Comparable<MobileSubscription>
Subscription
- name : String
+ Subscription(name : String, startDate : SMUDate)
+ equals(obj : Object) : boolean
+ getName() : String
+ getStartDate() : SMUDate
there is no getTalkTime method in this class.
MobileSubscription
- talkTime : int
+ MobileSubscription(name : String, startDate : SMUDate, talkTime : int)
+ compareTo(another : MobileSubscription) : int
+ equals(obj : Object) : boolean
+ toString() : String
NewspaperSubscription
- publisher : String
+ NewspaperSubscription(name : String, startDate : SMUDate, publisher : String)

APPENDIX B


Course
- code : String
- term : String
+ Course(code : String, term : String)
+ getCode() : String
+ getTerm() : String
+ equals(obj : Object) : boolean
+ hashCode() : int
+ toString() : String
Utility4
+ getCoursesTakenInSameYearAndSemesterOfStudy(courseList1 : List<Course>, courseList2 : List<Course>) : List<String>
+ main(args : String[]) : void
Two courses are the same if their code and term are the same
ExchangeCourse
+ ExchangeCourse(code : String, term : String, university : University)
+ getUniversity() : University
+ equals(obj : Object) : boolean
+ hashCode() : int
1
University
- name : String
- country : String
+ University(name : String, country : String)
+ getName() : String
+ getCountry() : String
+ hashCode() : int
+ equals(obj : Object) : boolean
+ toString() : String
Two ExchangeCourse are the same if their code, term and universities are the same
Two Universities are the same if they have the same name and country

- END OF PAPER -