COMP2026 Problem Solving Using Object Oriented Programming

Laboratory 10

Part A Discovery Exercises

Task 1: More about Constructors

a) What will be the output of the below program?

```
class A {
    public A() {
        System.out.println("Class A Constructor");
    }
}

class B extends A {
    public B() {
        System.out.println("Class B Constructor");
    }
}

class C extends B {
    public C() {
        System.out.println("Class C Constructor");
    }
}

public class MyMainClass {

    public static void main(String[] args) {
        C c = new C();
    }
}
```

Answer:

b) Explain why the following classes do not compile.

```
public class Papa {
    private int x;

    public Papa(int x) {
        this.x = x;
    }
}
```

```
public class Son extends Papa {
    public Son() {

    }

    public static void main(String[] args) {

    }
}
```

Answer:

c) Remove the constructor in **`Papa`** class. Explain why the following classes can compile now.

```
public class Papa {
    private int x;

}
```

```java
public class Son extends Papa {
    public Son() {

    }

    public static void main(String[] args) {

    }
}
```

Answer:

Task 2: Method Overriding

a) Create the following classes in an IntelliJ project.

```java
public class Person {

    private String name;

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

    public String getName() {
        return name;
    }

    public void saySomething() {
        System.out.println("I am " + name + ". I am a person.");
    }
}
```

```java
public class Student extends Person {

    private String major;

    public Student(String name, String major) {
        super(name);  //call the super class's constructor
        this.major = major;
    }
}
```

```java
public class Teacher extends Person {

    private double monthlySalary;

    public Teacher(String name, double monthlySalary) {
        super(name);  //call the super class's constructor
        this.monthlySalary = monthlySalary;
    }
}
```

```java
public class Tester {                                                    ⚠ 3

    public static void main(String[] args) {

        Person p = new Person( name: "Alan");
        Student s = new Student( name: "Bob",  major: "CS");
        Teacher t = new Teacher( name: "Chris",  monthlySalary: 1000);

    }
}
```

Run the **`Tester`** program to make sure it is error free before you move on to the next step.

b) Add the following **saySomething()** method calls in the **Tester** class.

```
public class Tester {

    public static void main(String[] args) {

        Person p = new Person( name: "Alan");
        Student s = new Student( name: "Bob",  major: "CS");
        Teacher t = new Teacher( name: "Chris",  monthlySalary: 1000);

        p.saySomething();
        s.saySomething();
        t.saySomething();

    }
}
```

c) Run the program again and paste the output below.

d) There is no **saySomething()** method in the **Student** and the **Teacher** class. Explain the output shown in part (c).

e) Add the following **saySomething()** method in the **Student** class.

```
public class Student extends Person {

    private String major;

    public Student(String name, String major) {
        super(name);  //call the super class's constructor
        this.major = major;
    }

    public void saySomething() {
        System.out.println("I am " + super.getName() +
                           ". I am a student. " +
                           "My major is " + major + ".");
    }
}
```

f) Run the program again and paste the output below.

g) Dose part (f) produce the same result in part (c). Why?

h) Add the following **`saySomething()`** method calls in the **`Teacher`** class.

```java
public class Teacher extends Person {

    private double monthlySalary;

    public Teacher(String name, double monthlySalary) {
        super(name);  //call the super class's constructor
        this.monthlySalary = monthlySalary;
    }

    public void saySomething(){
        super.saySomething();
        System.out.println("I am a teacher, too!");
    }
}
```

i) Run the program again and paste the output below.

j) Modify in the **`Tester`** class to assign the Student and Teacher objects to Person references.

```
public class Tester {
    public static void main(String[] args) {

        Person p = new Person( name: "Alan");
        Person s = new Student( name: "Bob",  major: "CS");
        Person t = new Teacher( name: "Chris",  monthlySalary: 1000);

        p.saySomething();
        s.saySomething();
        t.saySomething();
    }
}
```

k) Run the program again and paste the output below.

Task 3: ArrayList

The ArrayList class is a resizable array, which can be found in the java.util package. See https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html for more information.

The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever we want.

a) Write statement(s) to create an ArrayList object called **`fruits`** that will store strings.

b) Write statement(s) to add the following strings into the ArrayList.
"apple", "orange", "banana", "strawberry", "kiwi"

c) Write statement(s) to print the element with index 3 in the ArrayList.

d) Write statement(s) to remove element with index 2 in the ArrayList.

e) Write statement(s) to print all the element in the ArrayList.

Part B Programming Exercises

Task 1: Events

There are three kinds of events in a schedule book: **`OnetimeEvent`**, **`DailyEvent`** and **`MonthlyEvent`**. An event has and id, a description (e.g. "see dentist") and a date.

a) Write a class named **`OnetimeEvent`** that is a subclass of the given **`Event`** class. The class should have the following members:
   - A constructor that accepts the description, the year, month and day of the event as arguments.
   - Override the **`toString`** method to return the string representation in the format: **`description`**`(One time) (id: id)`

b) Write a class named **`DailyEvent`** that is a subclass of the given **`Event`** class. The class should have the following members:
   - A constructor that accepts the description of the event as argument.
   - Override the **`toString`** method to return the string representation in the format: **`description`**`(Daily) (id: id)`
   - Override the **`occursOn`** method to return an appropriate value. Note that daily event occurs every day.

c) Write a class named **`MonthlyEvent`** that is a subclass of the given **`Event`** class. The class should have the following members:
   - A constructor that accepts the description and the day of the month of the event as arguments. It is fine to create a monthly event that occurs on 29th, 30th, or 31st of the month. Assume that if a month doesn't have the given number of days, then there is no event that month.
   - Override the **`toString`** method to return the string representation in the format: **`description`**`(Monthly) (id: id)`
   - Override the **`occursOn`** method to return an appropriate value. Note that monthly event occurs every month on the specified day of the month.

Test your programs with the given **`ScheduleBookTester.java`**.

Sample outputs:

```
Enter the date(YYYY MM DD): 2030 5 2
See dentist(One time) (id: 1)
Visit Dad(Monthly) (id: 7)
Yoga class(Daily) (id: 9)
```

```
Enter the date(YYYY MM DD): 2030 12 5
Pay Bills(Monthly) (id: 6)
Yoga class(Daily) (id: 9)
```

```
Enter the date(YYYY MM DD): 2030 12 12
Yoga class(Daily) (id: 9)
```

Task 2: Schedule Book

Complete the given schedule book program **`Schedule.java`** that allows user to make inquiries, save the event data to a file and reload the data from a file.

a) Override the **`toFileString()`** method in the three event subclasses to return strings in the following formats. You may refer to `event.txt` for examples.
   - OnetimeEvent format: `Onetime` **`year month day description`**
   - DailyEvent format: `Daily` **`description`**
   - MonthlyEvent format: `Monthly` **`day description`**

b) Write a method called **`saveEvents(String filename)`** in the **`ScheduleBook`** class to save all the events in the array list into the specified file. The method should save the string returned by **`toFileString()`** into the file.

a) Write a method called **`loadEvents(String filename)`** to read event data from the file specified (e.g. events.txt), create the corresponding event objects and add the objects to the array list. The method should return the number of successfully loaded events.

b) Write a method called **`addEvent(Scanner in)`** in the **`ScheduleBook`** class to add an event to the array list according to the user input. See option 4 in the sample output.

c) Write a method call **`removeEvent(int id)`** in the **`ScheduleBook`** class to remove the event with the specified id.

d) Modify the **`runApp()`** method to allow user to select the operations from a menu.

Sample outputs:

```
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 1
Enter input filename: events.txt
8 event(s) successfully read.
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 3
Enter the date(YYYY MM DD): 2030 11 5
Visit Ada(One time) (id: 5)
Pay Bills(Monthly) (id: 6)
Yoga Class(Daily) (id: 8)
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit
```

```
Option: 5
Enter event id: 5
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 3
Enter the date(YYYY MM DD): 2030 11 5
Pay Bills(Monthly) (id: 6)
Yoga Class(Daily) (id: 8)
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 4
Event type (onetime, daily or monthly) : onetime
Enter the date(YYYY MM DD): 2030 11 15
Enter description: Assignment due
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit
```

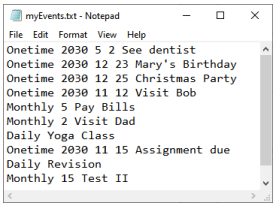
```
Option: 4
Event type (onetime, daily or monthly) : daily
Enter description: Revision
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 4
Event type (onetime, daily or monthly) : monthly
Enter the day of month: 15
Enter description: Test II
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 3
Enter the date(YYYY MM DD): 2030 11 15
Yoga Class(Daily) (id: 8)
Assignment due(One time) (id: 9)
Revision(Daily) (id: 10)
Test II(Monthly) (id: 11)
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit
```

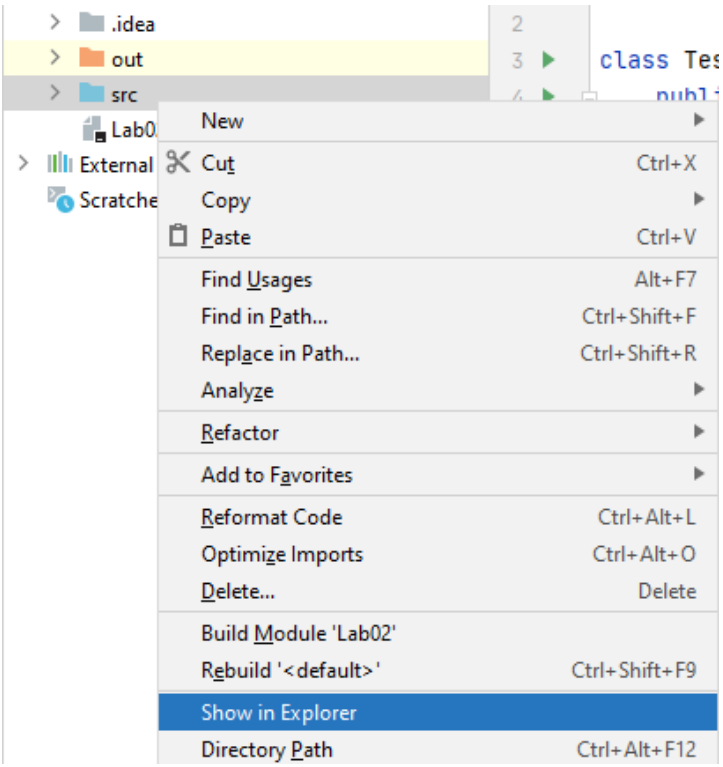
```
Option: 2
Enter output filename: myEvents.txt
1. Load events from file
2. Save events to file
3. Show events
4. Add event
5. Remove event
6. Quit

Option: 6
Bye~
```

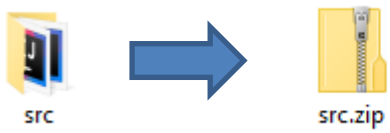


Part C Submitting Exercises

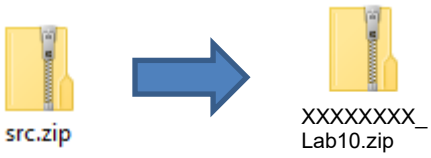
Step 1: Right-click the **src** folder and select **Show in Explorer**



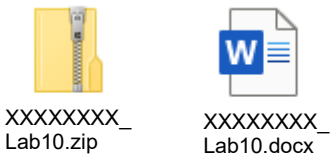
Step 2: Zip the **src** folder into **src.zip**



Step 3: Rename the src.zip file to **XXXXXXXX_lab10.zip** where **XXXXXXXX** is your **student id**



Step 4: Submit **XXXXXXXX_lab10.zip** and **XXXXXXXX_lab10.docx** to Moodle.



References

[1] Bravaco, R., & Simonson, C. (2009). *Java programming: From the ground up*. Dubuque, IA: McGraw-Hill.

[2] Dean, J., & Dean, R. (2008). *Introduction to programming with Java: A problem solving approach*. Boston: McGraw-Hill.

[3] Farrell, J. (2012). *Java programming. Boston, MA: Course Technology Cengage Learning*

[4] Forouzan, B. A., & Gilberg, R. F. (2007). *Computer science: A structured programming approach using C (3rd ed.)*. Boston, MA: Thomson Course Technology.

[5] Gaddis, T. (2016). *Starting out with Java (6th ed.)*. Pearson.

[6] Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.

[7] Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.

[8] Schildt, H., & Skrien, D. J. (2013). Java programming: A comprehensive introduction. New York: McGraw-Hill.

[9] Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education

[10] Xavier, C. (2011). *Java programming: A practical approach*. New Delhi: Tata McGraw Hill.

[11] yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming

[12] Zakhour, S., Kannan, S., & Gallardo, R. (2013). *The Java tutorial: A short course on the basics (5th ed.)*.