COMP2026 Problem Solving Using Object Oriented Programming

Laboratory 9

Part A Discovery Exercises

Task 1: Memory Model

a) Draw the memory model of the following code fragment and then write down the resulted values of all the variables.

```
int num1 = 21;
int num2 = 22;
int num3 = 23;
int[] a = {10, 11, 12, 13};
int[] b = {30, 31, 32, 33};

num3 = num1;
num1 = num2;
num2 = num3;

a[2] = 42;
b = a;
b[0] = 40;
```

Answer:

Results:

num1 = _____. num2 = _____. num3 = _____.

a = _____. b = _____.

b) Draw the memory model of the following code fragment.

```
Person a = new Person("Alan");
Person b = new Person("Bob");
Person c = new Person("Carlie");
a.setFriend(b);
b.setFrined(c);
c.setFriend(a);
```

```
public class Person {
    String name;
    Person friend;

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

    public String getName() {
        return name;
    }

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

    public Person getFriend() {
        return friend;
    }

    public void setFriend(Person p){
        this.friend = p;
    }
}
```

Answer:

Task 2: Public vs Private

a) Add the following statements to the **`main`** of the given **`Person.java`** to modify the name of object p.

```
public static void main(String[] args){
    Person p = new Person( name: "Alice",  age: 20);
    System.out.println(p);
    p.name = "Ada";
    System.out.println(p);
}
```

b) Run the program and paste the screenshot of the output below.

c) Add the given **`Group.java`** to the same project. Now, **`Person.java`** and **`Group.java`** are inside the **`src`** folder. Run the **`Group.java`** program and paste the screenshot of the output below.

d) Add the following statements to the given **`Group.java`** to modify the name of object p.

```
public static void main(String[] args){
    Person p = new Person( name: "Alice",  age: 20);
    System.out.println(p);
    p.name = "Ada";
    System.out.println(p);
}
```

e) Run the **`Group.java`** program and paste the screenshot of the output below.

f) Modify the **`Person.java`** to change the **`name`** field to public as follow.

```
public class Person {

    public String name;
    private int age;
```

g) Run the **`Group.java`** program again and paste the screenshot of the output below.

h) Modify the **`Person.java`** to change the **`name`** field back to private. Run the **`Group.java`** program again should result in error message.

```
public class Person {

    private String name;
    private int age;
```

i) Add the following statements to the **`main`** of the given **`Group.java`** to modify the name of object p.

```
public static void main(String[] args){
    Person p = new Person( name: "Alice",  age: 20);
    System.out.println(p);
    p.setName("Ada");
    System.out.println(p);
}
```

j) Run the **`Group.java`** program and paste the screenshot of the output below.

k) Remove the **`setName`** method in **`Person.java`**. Run the **`Group.java`** program again and paste the screenshot of the output below.

l) With private **`name`** field and no public **`setName`** method in **`Personjava`**. Can you modify the name of the Person object in **`Group.java`**?

_____.

Task 3: Writing our own Comparing Methods

Refer to the given **`Bee.java`**, answer the following questions.

a) Write a method called **`compareSpeed`** that compares the speed of this Bee with a specified Bee object. The method

- returns a negative integer if this Bee's speed is less than the speed of the specified Bee object
- returns a positive integer if this Bee's speed is greater than the speed of the specified Bee object
- returns 0 if the speeds are the same

Add the method into **`Bee.java`**.

```
public int compareSpeed(Bee obj){



}
```

b) Test your **`compareSpeed`** method with the following **`main()`** method in **`Bee.java`**.

```
public static void main(String[] args){
    Bee alice = new Bee( x: 55,  y: 25,  speed: 5);
    Bee bob = new Bee( x: 78,  y: 97,  speed: 3);

    if(alice.compareSpeed(bob) < 0){
        System.out.println("Alice flies slower than Bob.");
    }
    else if(alice.compareSpeed(bob)> 0){
        System.out.println("Alice flies faster than Bob.");
    }
    else {
        System.out.println("Alice and Bob fly in the same speed.");
    }
}
```

c) Run the program and paste the screenshot of the output below.

Part B Programming Exercises

Task 1: Phone Book Contact

A phone book contact holds a name and a list of phone numbers. Write a **`Contact`** class that keeps a name and an array of phone numbers.

The class contains:

- A private String field that holds the name of a person.
- A private String array that holds a list of phone numbers.
- A public constructor that constructs a contact with the specified name of a person. The array size of the phone number list should be set to zero.
- A public constructor that constructs a contact with the specified name of a person and a phone number of that person.
- A public **`getName`** method to return the name of the person.
- A public **`getPhoneNos`** method to return the array of phone numbers.
- A public **`addPhoneNo`** method to add a phone number into the array of phone numbers.
- A public **`deletePhoneNo`** method to remove the specified phone number from the array of phone numbers.
- A public method named **`toString`** that returns the string representation of the contact in the following format:

```
Name: name
Phone number(s):
[array index] phone number1
[array index] phone number2
...
[array index] phone numberN
```

For example, a contact of two phone numbers would return a String as follows.

```
Name: Chan Tai Man
Phone number(s):
[0] 91234567
[1] 97654321
```

Test your Contact class with the following **`main`** method:

```
public static void main(String[] args){
    Contact c1 = new Contact("Chan Tai Man", "91234567");
    System.out.println(c1);
    c1.addPhoneNo("97654321");
    System.out.println(c1);

    String [] p = c1.getPhoneNos();
    for(int i = 0; i < p.length; i++){
        System.out.println(p[i]);
    }

    c1.deletePhoneNo("91234567");
```

```
        System.out.println(c1);


        Contact c2 = new Contact("Luk Mei Mei");
        System.out.println(c2);

    }
```

Sample output:

```
Name: Chan Tai Man
Phone number(s):
[0] 91234567

Name: Chan Tai Man
Phone number(s):
[0] 91234567
[1] 97654321

91234567
97654321
Name: Chan Tai Man
Phone number(s):
[0] 97654321

Name: Luk Mei Mei
Phone number(s):
```

Task 2: Phone Book

A phone book holds a list of contacts. Write a class named **`PhoneBook`** to keep an array of contacts. The contacts belong to the class **`Contact`**. A phone book is empty when it is created. The phone book allows users to look up a name to find the associated phone number(s) and to make changes to the phone book. You could assume the names in the phone book are unique.

The class contains:

- A private **`Contact`** array field that represents the phone book.
- A public constructor that constructs an array of contacts with zero size.
- A public **`addContact`** method to add a **`Contact`** to the phone book in a way that the contacts in the array are **always sorted in lexicographical order by the name of the person**. The method should accept two arguments, the name of the person and a phone number. Note: You could use the **`compareTo()`** method in the String library to compare the names.
- A public **`getContact`** method to return the contact with the specified name of the person. The method should return **`null`** if the name of the person cannot be found in the phone book.
- A public **`addPhoneToExistingContact`** method to add a phone number to

an existing contact with specified name of the person.

- A public **`updateContact`** method to delete an old phone number and add a new phone number to an existing contact with specified name of the person.
- A public **`removeContact`** method to delete a contact with the specified name from the phone book.
- A public method named **`toString`** that returns the string representation of all the contacts in the phone book.

Note that you need to use the Contact class in Task 1.

Test your PhoneBook class with the following **`main`** method:

```
public static void main(String[] args) {
    PhoneBook b = new PhoneBook();
    //add contacts
    b.addContact("Chan Tai Man", "96385274");
    System.out.println(b);
    b.addContact("Ma Kin", "65478921");
    System.out.println(b);
    b.addContact("Au Siu Ming", "94562584");
    System.out.println(b);
    b.addContact("Koo Ka Ka", "91122334");
    System.out.println(b);

    //get contacts
    System.out.println(b.getContact("Chan Tai Man"));
    System.out.println(b.getContact("X X X")); //null
    System.out.println();

    //add phone no. to an existing contact
    b.addPhoneToExistingContact("Au Siu Ming", "61234578");
    System.out.println(b.getContact("Au Siu Ming"));

    //update the phone no. in an existing contact
    b.updateContact("Ma Kin", "65478921", "61231231");
    System.out.println(b.getContact("Ma Kin"));

    //remove contacts
    b.removeContact("Ma Kin");
    System.out.println(b);
    b.removeContact("Chan Tai Man");
    System.out.println(b);
    b.removeContact("Au Siu Ming");
    System.out.println(b);
    b.removeContact("Koo Ka Ka");
    System.out.println(b);
}
```

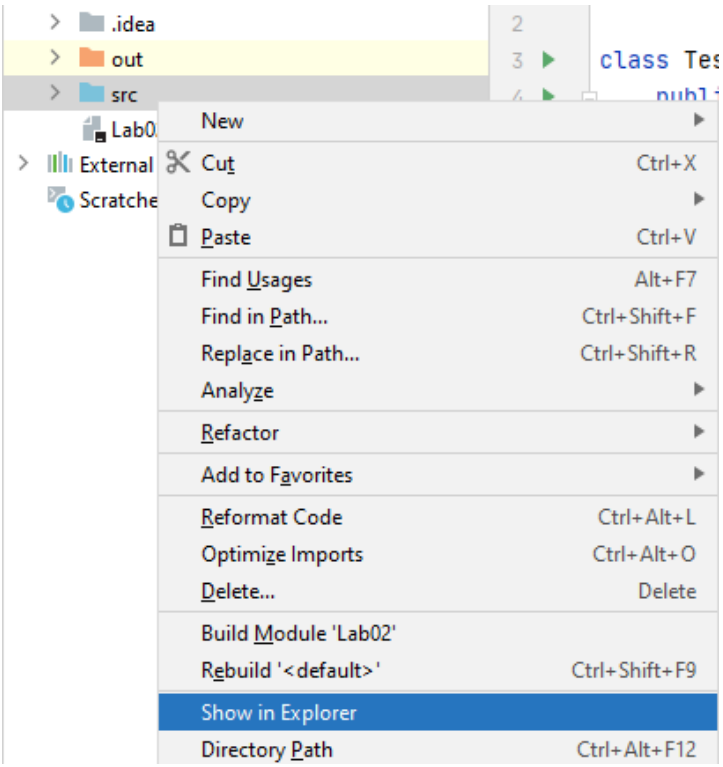
Sample output:

```
Name: Chan Tai Man
Phone number(s):
[0] 96385274

Name: Chan Tai Man
Phone number(s):
[0] 96385274
Name: Ma Kin
Phone number(s):
[0] 65478921

Name: Au Siu Ming
Phone number(s):
[0] 94562584
Name: Chan Tai Man
Phone number(s):
[0] 96385274
Name: Ma Kin
Phone number(s):
[0] 65478921

Name: Au Siu Ming
Phone number(s):
[0] 94562584
Name: Chan Tai Man
Phone number(s):
[0] 96385274
Name: Koo Ka Ka
Phone number(s):
[0] 91122334
Name: Ma Kin
Phone number(s):
[0] 65478921

Name: Chan Tai Man
Phone number(s):
[0] 96385274

null

Name: Au Siu Ming
Phone number(s):
[0] 94562584
[1] 61234578

Name: Ma Kin
Phone number(s):
[0] 61231231
```

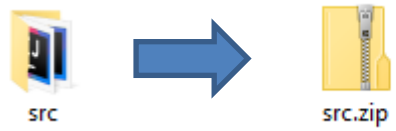
```
Name: Au Siu Ming
Phone number(s):
[0] 94562584
[1] 61234578
Name: Chan Tai Man
Phone number(s):
[0] 96385274
Name: Koo Ka Ka
Phone number(s):
[0] 91122334

Name: Au Siu Ming
Phone number(s):
[0] 94562584
[1] 61234578
Name: Koo Ka Ka
Phone number(s):
[0] 91122334

Name: Koo Ka Ka
Phone number(s):
[0] 91122334
```

Part C Submitting Exercises

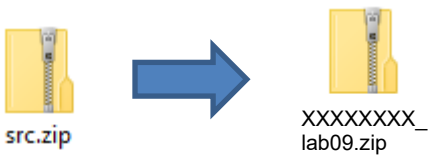
Step 1: Right-click the **src** folder and select **Show in Explorer**



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



Step 3: Rename the src.zip file to ***XXXXXXXX_lab09.zip*** where ***XXXXXXXX*** is your **student id**



Step 4: Submit ***XXXXXXXX_lab09*.zip** and **XXXXXXXX_lab09.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.)*.