COMP2026 Problem Solving Using Object Oriented Programming

# Laboratory 10

**Part A Discovery Exercises**

**Task 1: More about Constructors**

1. What will be the output of the below program?

|  |
| --- |
|  |

Answer:

|  |
| --- |
| Class A Constructor  Class B Constructor  Class C Constructor |

1. Explain why the following classes do not compile.

|  |
| --- |
|  |

|  |
| --- |
|  |

Answer:

|  |
| --- |
| Because ‘Papa’ don’t have a default constructor |

1. Remove the constructor in **Papa** class. Explain why the following classes can compile now.

|  |
| --- |
|  |

|  |
| --- |
|  |

Answer:

|  |
| --- |
| Removed the constructor ,then no need to use default constructor in “Papa” class |

**Task 2: Method Overriding**

1. Create the following classes in an IntelliJ project.

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

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

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

|  |
| --- |
|  |

1. Run the program again and paste the output below.

|  |
| --- |
| I am Alan. I am a person.  I am Bob. I am a person.  I am Chris. I am a person. |

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

|  |
| --- |
| Because the **Student** and the **Teacher** class Inheritance the Person class including inherits data (instance variables)and behavior (methods) from a superclass |

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

|  |
| --- |
|  |

1. Run the program again and paste the output below.

|  |
| --- |
| I am Alan. I am a person.  I am Bob. I am a student. My major is CS.  I am Chris. I am a person. |

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

|  |
| --- |
| No. Because class Student create the saySomething() by his own, and also replace the super method with the same method name |

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

|  |
| --- |
|  |

1. Run the program again and paste the output below.

|  |
| --- |
| I am Alan. I am a person.  I am Bob. I am a student. My major is CS.  I am Chris. I am a person.  I am a teacher, too! |

1. Modify in the **Tester** class to assign the Student and Teacher objects to Person references.

|  |
| --- |
|  |

1. Run the program again and paste the output below.

|  |
| --- |
| I am Alan. I am a person.  I am Bob. I am a student. My major is CS.  I am Chris. I am a person.  I am a teacher, too! |

**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.

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

|  |
| --- |
| List<String> fruits = new ArrayList<>(); |

1. Write statement(s) to add the following strings into the ArrayList.

“apple”, “orange”, “banana”, “strawberry”, “kiwi”

|  |
| --- |
| fruits.add(“apple”);  fruits.add(“orange”);  fruits.add(“banana”);  fruits.add(“strawberry”);  fruits.add(“kiwi”); |

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

|  |
| --- |
| String fruit = fruits.get(3);  System.out.println(fruit); |

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

|  |
| --- |
| fruits .remove(2); |

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

|  |
| --- |
| for (int i = 0; i < fruits.size();i++)  {  System.out.println(fruits.get(i));  } |

**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.)*.