|  |
| --- |
| **Software Engineering Department - ITU** |
| **SE102BL: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 20/02/2024** |
| **Teaching Assistant: Zain ul Abidin, Muhammad Abdullah** | **Semester: Spring 2024** |
| **Lab Engineer: Muhammad Saqib Baig** | **Batch: BSSE2023(Section B)** |

# **Lab 4B. Solving Problems by using Inheritance**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| Saleh bint e bilal | Bsse23081 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to practice problems related to pointers and dynamic memory allocation.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Inheritance:**

Inheritance is a fundamental concept in object-oriented programming (OOP) that allows one class (the derived or child class) to inherit the properties and behaviors of another class (the base or parent class). This relationship is established to promote code reuse, extend existing functionality, and create a hierarchical structure within the program.

**Base Class:**

* Also known as the parent or superclass, the base class is the class that provides the common properties and behaviors to one or more derived classes.
* It serves as a blueprint for creating objects and defines the fundamental characteristics that are shared by all its derived classes.
* The base class encapsulates common functionality, promoting code reusability and maintaining a modular and organized code structure.

**Derived Class:**

* Also known as the child or subclass, the derived class inherits properties and behaviors from the base class.
* It can extend or override the functionality of the base class, adding specific features or modifying existing ones.
* The derived class can have its own unique members in addition to those inherited from the base class.

**Syntax for inheritance:**

**class BaseClass {**

**// Base class members and functions**

**};**

**class DerivedClass : public BaseClass {**

**// Derived class members and functions**

**};**

**Types of Inheritance:**

* **Single Inheritance:** A class can inherit from only one base class.
* **Multiple Inheritance:** A class can inherit from multiple base classes.
* **Multilevel Inheritance:** A class can inherit from a base class, and another class can inherit from the derived class, forming a chain.

**Lab Tasks**

**Task 1: Book Class Implementation**

**1.** **Class Definition (‘Book’):**

Create a base class ‘**Book**’. Book is a base class representing generic information about a book, with ‘**title**’, ‘**author**’ and ‘**year**’ as public members.

* **‘Book**’ class has members title (string), author (string), and year (int). Constructor initializes these members.
* ‘**FictionBook**’ and ‘**NonFictionBook**’ classes inherit from ‘Book’ and add additional members (‘genre’ and ‘subject’).

**2. Constructors:**

* ‘**Book**’ constructor initializes ‘**title**’, ‘**author**’, and ‘**year**’.
* ‘**FictionBook’** constructor initializes ‘**title**’, ‘**author**’, ‘**year**’, and adds ‘**genre**’.
* ‘**NonFictionBook**’ constructor initializes ‘**title’**, ‘**author**’, ‘**year’**, and adds ‘**subject**’.

**3. Base and Derived Classes:**

* ‘**FictionBook** ` is a derived class of ‘**Book**’ and ‘**NonFiction**’is another derived class of ‘**Book**.

**4. Member Functions:**

* **‘Book**’ class has a member function ‘**printInfo()**’ that prints information about the book.
* **‘FictionBook**’ class has a member function ‘**printGenre()**’ to print the genre of the fiction book.
* ‘**NonFictionBook**’ class has a member function ‘**printSubject()**’ to print the subject of the non-fiction book.

**Task 2: ElectronicDevice Class Implementation**

**1.** **Class Definition and Data Types:**

* **‘ElectronicDevice**’ class has members brand (string), model (string), and year (int). Constructor initializes these members.
* **Smartphone**, **Laptop**, and **Tablet** classes inherit from ‘**ElectronicDevice**’ and add additional members (**operatingSystem**, **ramSize**, and **displayType**).

**2. Constructors:**

* ‘**ElectronicDevice**’ constructor initializes ‘**brand**’, ‘**model**’, and ‘**year**’.
* **Smartphone** constructor initializes ‘**brand’**, ‘**model’**, ‘**year’**, and adds ‘**operatingSystem’**.
* **Laptop** constructor initializes ‘**brand’**, ‘**model’**, ‘**year’**, and adds ‘**ramSize’**.
* **Tablet** constructor initializes ‘**brand’**, ‘**model’**, ‘**year’**, and adds ‘**displayType**’.

**3. Base and Derived Classes:**

* ‘**ElectronicDevice**’ is the base class with members ‘**brand**’, ‘**model**’, and ‘**year**’.
* **‘Smartphone**’ is a derived class from ‘**ElectronicDevice**’ with an additional member ‘**operatingSystem**’.
* **Laptop** is another derived class from ‘**ElectronicDevice’** with an additional member ‘**ramSize**’.
* **‘Tablet’** is another derived class from ‘**ElectronicDevice**’ with an additional member ‘**displayType**’.

**4. Member Functions:**

* ‘**ElectronicDevice**’ class has a member function ‘**displayInfo( )**’ to print information about the electronic device.
* **‘Smartphone**’ class has a member function ‘**showOperatingSystem( )** to print the operating system of the smartphone.
* **Laptop** class has a member function ‘**displayRamSize( )’** to print the RAM size of the laptop.
* **Tablet** class has a member function ‘**showDisplayType( )**’ to print the display type of the tablet.

|  |
| --- |
|  |

Assessment Rubric for Lab

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance metric** | **CLO** | **Able to complete the task over 80% (4-5)** | **Able to complete the task 50-80% (2-3)** | **Able to complete the task below 50% (0-1)** | **Marks** |
| 1. Realization of experiment | 1 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed . | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed . | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non- existent. No testing has been completed. |  |
| 1. Conducting experiment | 1 | Able to make changes and answered all questions. | Partially able to make changes and few incorrect answers. | Unable to make changes and answer all questions. |  |
| 1. Computer use | 2 | Document submission timely. | Document submission late. | Document submission not done. |  |
| 1. Teamwork | 3 | Actively engages and cooperates with other group member(s) in effective manner. | Cooperates with other group member(s) in a reasonable manner but conduct can be  improved. | Distracts or discourages other group members from conducting the experiment |  |
| 1. Laboratory safety and disciplinary rules | 3 | Code comments are added and does help the reader to understand the code. | Code comments are added and does not help the reader to understand the code. | Code comments are not added. |  |
| 1. Data collection | 3 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap. | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables. | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy. |  |
| 1. Data analysis | 4 | Solution is efficient, easy to understand, and maintain . | A logical solution that is easy to follow but it is not the most efficient. | A difficult and inefficient solution . |  |
| **Total (out of 35):** | | | | |  |