

**FACULTY OF INFORMATICS**

|  |  |  |  |
| --- | --- | --- | --- |
| **SUBJECT’S INFORMATION:** | | | |
| Subject: | CSCI204 Object and Generic Programming | | |
| Session: | July 2014 | | |
| Programme / Section: | J766SENG (SE) / J766CS53 (MGD) / J766CS42 (DSS) | | |
| Lecturer: | Ms. Siti Hawa | | |
| Coursework Type  *(tick appropriate box)* | ✓ Individual Assignment ❑ Group Assignment ❑ Project  ❑ Lab Task ❑ Seminar / Tutorial Paper ❑ Others | | |
| Coursework Title: | **Assignment 2** | Coursework Percentage: | 15% |
| **ASSESSMENT CRITERIA:** | | | |
| Correctness | All programs should produce the correct result as stated in the specification. | | |
| Class Design and Implementation | Class declared and implemented as specified complete with proper validation on data. Appropriate data hiding and encapsulation implemented. | | |
| Main Function | Good design of the main function with complete functionalities included. Well-structured and used modular approached. Necessary data validation is implemented. | | |
| Readability | Appropriate comments are included. Meaningful identifiers used. Proper indentation and line spacing used. | | |
| Well formatted output | Output should be well formatted with appropriate messages displayed. Numbers are shown with appropriate precision. | | |
| **SUBMISSION:** | | | |
| You are to submit through Moodle:   1. The softcopy of the project containing all files (.cpp and .h). 2. A word document containing the UML class diagram with explanation for each class and relationship.   **SUBMIT AS EARLY AS POSSIBLE. YOU CAN RE-SUBMIT LATER IF NECESSARY. ONLY THE LATEST SUBMISSION WILL BE MARKED.**  **IF YOU SUBMIT YOUR ASSIGNMENT TWICE, ONE SUBMMISSION BEFORE THE DUE DATE AND ANOTHER AFTER THE DUE DATE, THEN YOU WILL BE PENALIZED FOR LATE SUBMISSON.** | | | |
| DUE DATE: | **Thursday, 16th October 2014, 5 pm** | | |
| **PENALTIES FOR LATE SUBMISSION:** | | | |
| Penalties apply to all late work, except if student academic consideration has been granted. Late submissions will attract a penalty of 25% of the assessment mark per day including the weekend. Work more than (3) days late will be awarded a mark of zero. | | | |
| **PLAGIARISM:** | | | |
| **When you submit an assessment task, you are declaring the following**   1. It is your own work and you did not collaborate with or copy from others. 2. You have read and understand your responsibilities under the University of Wollongong's policy on plagiarism. 3. You have not plagiarised from published work (including the internet). Where you have used the work from others, you have referenced it in the text and provided a reference list at the end ot the assignment.   Plagiarism will not be tolerated. Students are responsible for submitting original work for assessment, without plagiarising or cheating, abiding by the University’s policies on Plagiarism as set out in the University Handbook under University Policy Directory and in Faculty handbooks and subject guides. | | | |

**COURSEWORK SPECIFICATION**

**OBJECTIVES:**

This assignment will provide you with experience in writing C++ programs with

* Overloaded operators and friends
* Inheritance and Polymorphism

**Remember that:**

**1. All programs should be able to run on the lab’s computers.**

**2. You must put the following information on the header of each text and source file you will be submitting in this assignment:**

**Student’s full name:**

**Student’s ID:**

**Modification Date:**

**Purpose of this file (or program):**

**3. Assignments that are not able to be compiled will result in zero mark given to the assignment.**

**4. You must only use the C++ features that have already been covered in the lectures**

**TASK 1:**

Develop an application using a BankAccount class for a local bank. Include account number and balance as the data members and also suitable constructors and set/get functions. Include also the following overloaded operators to your class:

* The insertion operator (<<) to display a bank account’s information.
* The += operator that takes a double parameter to represent a deposit amount to be added to the BankAccount balance.
* The –= operator that takes a double parameter to represent a withdrawal amount to be deducted from the BankAccount balance.
* The > and < operator that determine whether one account is less than or greater than another based on the balance amount.
* The == operator that compares two BankAccount objects based on the account number.

Design and implement a driver program that allows the user to perform various operations on several BankAccount objects stored in an array. New BankAccount objects should be created with a minimum deposit of RM100.00. Make sure proper validations are done for each deposit or withdrawal made to an account. A new BankAccount created should not have the same account number with any of the existing BankAccount.

**TASK 2:**

Radio Frequency Identification (RFID) chips are small tags that can be placed on a product. They behave like wireless barcodes and can wirelessly broadcast an identification number to a receiver. One application of RFID chips is to use them to aid in the logistics of shipping freight. Consider a shipping container full of items. Without RFID chips, a person has to manually inventory all of the items in the container to verify the contents. With an RFID chip attached to the shipping container, the RFID chip can electronically broadcast to a human the exact contents of the shipping container without human intervention.

To model this application, write a base class called ShippingContainer that has a container ID number as an integer. Include member functions to set and access the ID number. Add a virtual function called getManifest that returns an empty string. The purpose of this function is to return the contents of the shipping container.

Create a derived class called ManualShippingContainer that represents the manual method of inventorying the container. In this method, a human simply attaches a textual description of all contents of the container. For example, the description might be “4 crates of apples, 10 crates of pears.” Add a new class variable of type string to store the manifest. Add a function called setManifest that sets this string. Override the getManifest function so that it returns this string.

Create a second derived class called RFIDShippingContainer that represents the RFID method of inventorying the container. To simulate what the RFID chips would compute, create an add function to simulate adding an item to the container. The class should store a list of all added items (as a string) and their quantity using the data structures of your choice. For example, if the add function are invoked three times as follows:

*rfidContainer.add(“crate of pears”); //add one crate of pears*

*rfidContainer.add(“crate of apples”); //add one crate of apples*

*rfidContainer.add(“crate of pears”); //add one crate of pears*

At this point, the data structure should be storing a list of two items: crate of apples and crate of pears. The quantity of apples is one and the quantity of pears is two. Override the getManifest function so that it returns a string of all items that is built by traversing the list of items. In the above example, the return string would be “2 crate of pears, 1 crate of apples.”

Include other necessary functions, constructors, and destructor to all the classes above.

Finally, write a main program that creates an array of pointers to 6 ShippingContainer objects. The array should be used to store both the ManualShippingContainer objects and RFIDShippingContainer objects. Construct the main program to allow the user to decide the number of each type of objects to be stored in the array. Instantiate all the objects accordingly and demonstrate how each objects behaves differently when the same instructions are given to them. For the ManualShippingContainer objects, you will have to invoke setManifest function to set the contents. For the RFIDShippingContainer objects, you will have to invoke add to set the contents (although, if this were real, the contents of the container would “add” themselves via the RFID chips instead of requiring a human to type them in).

**MARKS ALLOCATION:**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Marks Allocated** | |
| **Task 1** | **Task 2** |
| Correctness | 1 | 1 |
| Class Design and Implementation | 2 | 4 |
| Main Function | 1 | 3 |
| Readability | 0.5 | 0.5 |
| Well formatted output | 1 | 1 |