**Advanced Object Oriented Programming**

**Mini Assignment 2**

**(10%)**

**Average Time to complete: 2 and a half hours**

**Time given: 1 Week and 3 days.**

**Due:** Friday 15th October at 11:30 pm

**Submission:** On Blackboard.

Format: Upload this completed document

This project must be done in a group. **Group size: 2 to 4 people**.

It consists of 3 parts.

|  |  |  |  |
| --- | --- | --- | --- |
| **Group member** | **First name** | **Last name** | **Student id number** |
| **1** | Michael | Noseworthy | 101027533 |
| **2** | Ronal | Rodriguez Arias | 101314540 |
| **3** | Ryan | Lee | 101296633 |
| **4** |  |  |  |

**One member of the group is responsible for uploading the completed document.**

Question1:

What is inheritance with respect to OOP? (You must mention appropriate terminology)

In your explanation you must:

* Explain how it is implemented in the C# programming language
  + This includes constructor parameter passing and keywords
* Explain how it is represented in a UML diagram.

Total for question: [**4 marks**]

(Delete the lines after the question and insert your response)

Inheritance is the way one class inherits the attributes and methods from another class. The class in which is being inherited is called the parent class and the class that is inheriting is called the child class. When inheriting from a class your constructor must have parameters to pass the parents variables. You must use the base () keyword and in between the brackets there must be the variables that are passing through from the parent class’s constructor. Inheritance is represented with a white arrow.

Question 2:

What is polymorphism with respect to OOP? In your explanation you must:

Briefly explain the two main forms and any requirements they may have. You must also indicate how the forms are identified in a UML diagram**.**

Total for question: [**6 marks**]

(Delete the lines after the question and insert your response)

Polymorphism is the term used to describe an object having multiple forms. It deals and defines methods and their behaviors. The two types that polymorphism usually take are called overloading (compile time polymorphism) and overriding (runtime polymorphism). Overloading does not require inheritance. Furthermore, method signatures must be different, and names must be the same. With overloading, you can pass values into the parameters and the program will choose the method that satisfies the parameters entered. The methods could have a similar name, but the program will still choose the appropriate method regardless. In UML diagrams, overloading can be shown/identified in classes that contain methods with the same name in them. Overriding requires inheritance, method names and signatures must be identical. In a UML diagram, overriding is identified with the hollow diamond inheritance relationship line and the method name + signatures written similarly in the super class and the sub classes.

Question 3: [10 marks]

Draw a UML Class Diagram representing the following elements from the problem domain for a soccer league. A soccer league is made up of at least four soccer teams. Each soccer team is composed of seven(7) to eleven(11) players, and one player captains the team. A team has a name and a record of wins and losses. Players have a number and a position. Soccer teams play games against each other. Each game has a score and a location. Teams are sometimes lead by a coach. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses. Draw a class diagram for this information, and be sure to label all associations with appropriate multiplicities

All collections must be shown as arrays.

Carry out an initial object-oriented design for the above specification, in which you should identify and show in a UML diagram:

* Classes that you think will be required.
* Their attributes and behaviours
* Any aggregation relationships
* Any inheritance relationships you can identify
* Any other relationships between your classes

The UML Diagram should show all the information of the classes. No coordinator class is required.

**Please note that you must show or display competency in the techniques taught in the classes.**

**Proper usage of concepts such as manager classes, coordinator classes, aggregation and responsibility delegation must be displayed in your design.**

**The names and ID numbers of all group members must be in the diagram.**

**You can use the website “draw.io” to do your design and export the image. This was shown in the recorded class.**

Insert your UML diagram on the next page

**Insert your UML diagram after this line.**

Graphical user interface, application

Description automatically generated

CAREFUL NOTE:

- Please safeguard your own team’s work.

- You are to do this project with your group members only.

- Note: Mobi Help members are NOT supposed to do or help you with your assignment.