L23

Thursday, January 27, 2022 12:17 AM

OOP Concepts & Class Fundamentals

Object Oriented Programming Concepts

Q1. A **programming paradigm** defines the methodology of designing and implementing programs using the key features and other building blocks of a programm ing language.

A **programming paradigm** gives you an idea how problems are generally analyzed and solved in a particular programming language.

Below are a few commonly used programming paradigms :
1. Procedural Programming
2. Object Based Programming
3. Object Oriented Programming

Procedural programming lays more emphasis on **procedure** than **data. C** language uses procedural programming style.

C++ and **Java** can also be used as a procedural programming language with some enhanced features such as **type checking, reference variables, inline functions, default arguments** etc.

Below are a few **pros** and **cons** of procedural programming languages:

Pros
1. For smaller programs the code is usually small, more readable and result roriented.
2. They are usually very efficient since they are written for a very specific purpose.

Cons
1. Since programs are function based they do not relate to a real life object.
2. When large amount of code is written inside functions, it becomes very difficult to maintain or make modifications.
3. Changing the way data is stored can required large amount of changes in code, unlike in Object oriented programs, making it difficult to maintain large systems.

Select all the correct statements given below.

- [x] A programming paradigm informs how problems are **analyzed** and **solved** in a programming language.
- [ ] Procedural programming lays more emphasis on **data** and **procedures**.
- [x] Large systems developed using procedural programming language can be difficult to maintain.

Q2. Understanding Object Based and Object Oriented Progamming

Object based programming is a paradigm that implements some features of object oriented programming but not all.

In object based programming, data and its associated meaningful functions are enclosed in one single entity called class.

Classes enforce information hiding and abstraction thereby separating the implementation details.

In object based programming, whenever there is any **change** in the definition of type, user's interface remains unaffected generally.

The object based programming can be thought of as a **subset** of object oriented programming as it implements some of the features implemented by object oriented programming like **information hiding**, **abstraction**, **classes**, **function overloading**, etc.

The object based programming implements **inheritance** and **polymorphism**.

The advantages of object based programming are:
- it overcomes most shortcomings of procedural programming
- it localizes changes and hides implementation details from user
- it supports user-defined types
- implements information hiding and abstraction

However, object based programming suffers from one major **limitation** and that is its inability to represent real world relationships that exist among object.

Object oriented programming paradigm is a superset of object based programming. It offers all the features of object based programming and overcomes its limitation by implementing **inheritance**.

Select all the correct statements given below.

- [x] A class is a combination of data and its associated meaningful functions.
- [x] Object based programming implements information hiding and abstraction.
- [ ] Object oriented programming does not implement inheritance and polymorphism.

Q3. Understanding OOP Concepts

The **object oriented programming (OOP)** approach views a problem in terms of **objects** or **entities** involved rather than the procedures to be implemented.

For instance, a BMW X5 car is a real life **object**, whose **characteristics** are **color**, **top speed**, **number of seats**, etc. We can consider the car instance (also called object) to exhibit **behaviors** like: **start**, **accelerate**, **break**, etc.

While programming using **OOP** approach, the **characteristics** of an object are represented by its **data** and its **behavior** is represented by its **functions**.

Therefore, in object oriented programming object represents an entity that can store **data** and has its interface with external world through **functions**.

Another important term in object oriented programming is a class, which is actually a representation of **characteristics** and **behavior** of the same category (or family) of **objects**. Hence, an **object**, BMW X5, can be considered as an instance of a **class** called Car.

Select all the correct statements given below.

- [x] A class is a combination of data members and member functions.
- [x] An **object** is an instance of a **class**.
- [x] Data in an object captures the characteristics (or attributes) of that object.

Q4. Understanding Data abstraction and encapsulation

The object oriented programming evolved to overcome the **limitations** of procedural programming approach.

The fundamental concepts in **OOP** are:

1. Data Abstraction
2. Data Encapsulation
3. Modularity
4. Inheritance
5. Polymorphism

Data abstraction refers to the act of representing essential features without including the background details and explanations.

Let us consider an example of **switch board**, you can only press certain switches according to your requirement. What is happening inside, how it is happening, you need not know.

This is **abstraction**, you know only essential things to operate on **switch board** without knowing the background details of switch board.

Data encapsulation is the way of combining both data and functions that operate on the data under a single unit.

In this, the **data** is not accessed to the outside world directly and only through member functions, the data can be accessed.

The insulation of data from direct access by the program is called **data hiding**.

Abstraction and encapsulation are complementary concepts, where as **abstraction** focuses upon the observable behavior of an object and **encapsulation** focuses upon the implementation that gives rise to this behavior.

Encapsulation is most often achieved through **information hiding**, which is the process of hiding all the secrets of an object that do not contribute its essential characteristics, the structure of an object is hidden as well as the implementation of its methods.

Select all the correct statements given below.

- [x] Abstraction is representing only essential things without knowing the background details.
- [ ] A class contains only member functions to secure the data.
- [x] Encapsulation is combining both data and functions.

Class Fundamentals

Q1. What is a Class?

According to English dictionary, a class is defined as ***a set*** *or category of* ***things*** *having* ***some property*** *or attribute* ***in common***.

For example, we can consider a banana, a mango and an orange to belong to a class called fruit.

In an object-oriented programming language like Java, a class acts as a type, a **template** a blueprint, from which instances of that type can be created.

Below is the syntax for a simple class called Student:

```
public class Student {
}
```

1. The word Student is the name (identifier) of the class.
2. The keyword class qualifies/tags this identifier Student as a **class**.
3. The word public is called access modifier, it is used to control access. We will learn more about access modifiers in the ensuing sections.

A class can contain members such as fields, methods, nested classes, interfaces, instance, static initializers and constructo rs.

In Java, String is a pre-defined class representing a sequence of characters.
So when we say :

```
String text1 = new String("Ganga");
String text2 = new String("River");
```

we are creating two instances, namely, text1, text2 of type String which hold "Ganga" and "River" character sequences respectively.

Select all the correct statements given below.

- [ ] A class cannot contain another class.
- [ ] A class need not contain the keyword `class` in its declaration.
- [ ] `int` and `Integer` are predefined classes in Java.
- [x] In the statement:

  ```
  Integer age = new Integer(25);
  ```

  `age` is an instance of class Integer.

Q2. Class and its Fields

If we consider a vehicle, say a car, we would immediately think of its model, colour, maximum speed etc. These are the variou s attributes also called properties of an instance of a car whose values could vary from model to model.

Similarly, when we consider a class say, Student, we can list the most common attributes/properties of a Student class as id, name, age, gender.

Note the spellings of id, name, age and gender they are written using camel-case convention and they all start with a lower case letter.

private is an access modifier like public and it is a Java keyword written in lower case.

int and char are primitive types written in lower case.
String is a class name and hence is written with the starting letter in upper case.

See the code and retype the same.

q11154/Student.java | q11154/StudentMain.java

```
1  package q11154;
2  public class Student {
3      private String id;
4      private String name;
5      private int age;
6      private char gender;
7  }
```

Q3. Student Class with Fields

In a class the fields are usually declared with private access modifier. We will learn more about access modifiers in the ensuing sections.

A field declaration could also contain initialization. For example:

```
private int age = 25;
```

Field declarations of similar types can be done as a single statement. For example:

```
private int age = 25, totalMarks = 100;
```

Note that every statement must be terminated by a semicolon.

Identify the errors and correct them.

q11155/Student.java | q11155/StudentMain.java

```java
package q11155;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;
}
```

Q4. Identify the errors in below code and correct them.

q11217/Student.java | q11217/StudentMain.java

```java
package q11217;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;
}
```