Final Assignement
Fall 2020-21

Course: Object Oriented Programming (1) / Java

Section: L

Prepared By:
Saiydur Rahman
19-39890-1

Submitted to:
MD. NAZMUL HOSSAIN
Lecturer, CS

Answer of Question No: 1

Briefly Explanation of following Key word

a

Abstract: "Abstract" the name is come from abstraction. Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achived with abstract class and abstract method.

abstract keyword is a non-access modifier.

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class)

Abstract method: ~~cant~~ can only used in an abstract class, and it does not have a body. The body is provided by the subclass.

b

Final: The final keyword is used to restrict the user. It can be used in many context.

(I) variable (II) method (III) class

it can be initialized only in construcor. We used final key word to stop

(I) Value change (Make it constant)
(II) Stop method Overriding
(III) Stop inheritance

(c) **Interface**: An interface is a blueprint of a class. It has static constants and abstract methods. The interface is a mechanism to achive abstraction. Interface is a class where can be only abstract methods not method body. Interface represent the Is-A relationship

We use interface because

(i) It is used to achive abstraction

(ii) By interface, we can support the functionality of multiple inheritance.

(iii) It can be used to achive loose ~~of~~ coupling.

(d) **Package**: A java package is a group of similar types of classes, interface and sub. packges. Package in java can be categorized in two form, built in package and user defined package. There are some advantages of package

(i) Package is used to categorized the clames and ~~int~~ inheritance so that they can be easily maintained

(ii) Java package provides access protection.

(iii) Java package removes naming collisions.

(6) **super:** super is a keyword in java. It is a reference variable which is used to refer immediate parent class object. Whenever we create the instance of subclass, an instance of parent class is created implicitly which referred by super reference variable.

(i) super can be used to refer immediate parent class.

(ii) super can be used to invoke immediate parent class method.

(iii) super() can be used to invoke immediate parent class constructor.

(7) **throws:** throws keyword is used to declare ~~and~~ an exception. It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained. It will use in a method start or class method start.

(g) try: A try statement is used to catch exceptions that might be thrown as program executes. Whenever we use a statement that might thrown an exception that way our program won't crash if the exception occurs. try statement is a block method first write try keyword then in the block write that code which can be throws exception on occurs.

(h) catch: In catch block we handle the exceptions. this block must follow the try block. A single try block can have several catch block associated with it. We can catch different exceptions in different catch block. When an exception occurs in try block, the corresponding catch block that handles the particular exceptions executes.

(i) extends: An object can acquire the properties and behaviour of the another object using Inheritance. The extends keyword extends a class (indicates that a class is inherited from another).
a subclass(child) extends ~~supp~~ supper class (parent class)

(j) **implements** : implements keyword is used to implement an interface. The interface keyword is used to declare a special Type of class that only contain abstract methods. To access the inteface methods the interface must be implemented by another class with implements keyword.

(k) **import** : import statement is used to bring certain classes or the entire packages into visibility. As soon as imported, a class can be referred to directly by using only its name. Use import to access built-in and user-defined packages into java source file so that the class can refer to a class that is in another package by directly using its name.

(l) **null** : null in java is a literal of the null type. It cannot be cast to a primitive type such as int, float etc. but can be cast to a reference type. Also, null does not have the value 0 necessarily.

(m) **GUI**: Graphical User Interface in java an ~~easy~~ easy-to-use visual experience builder for Java applications. It is mainly made of graphical components like buttons, labels, windows, etc. Through which the user can interact with an application. GUI plays and important role to build easy interfaces for Java applications.

(n) **finally**: finally defines a block of code we use along with the try keyword. It defines code that's always run after the try and any catch block but before the method is completed. We generally use the finally block to execute clean up code like closing connections, closing file, or freeing up threads as it executes regardless of an exception.

Answer to the Question No: 2

(a) OOP means Object Oriented Programming which contains Four features. If there have four features in a programming then it will be called OOP. The four main feature one

(I) Abstraction — Data Hiding
(II) Encapsulation — Making things private/protected
(III) Inheritance — Relationships
(IV) Polymorphism — Override/Overload

If we consider a Mobile phone as a OOP then we will get this four things like a mobile has IMEI number and many hidden things which we cannot see but we use it. Every IMEI NUMBER is private or different from other mobile. There have many types of mobile like Android, Symbian but they all are mobile which is ~~mob~~ inheritance. And we use many things from mobile which all are same but in diffirent look.

Page:8

(b) Encapsulation is one of the main concept or fundamental in OOP. Encapsulation in Java is a mechanism of wrapping the data and code acting on the data together as a single unit. In encapsulation, the variables of a class will be hidden from other class and can be accessed only through the methods of thier class. That's why it is also as data hiding. To achive encapsulation in java need to declare variable as a private. make setter getter methods to modify and view the variables. There are some benifits of encapsulation

(i) The fields of a class can be made read only or write only

(ii) A class can have ~~to to~~ total control over what is stored in its field.

© To achive encapsulation in Java we need to
(i) declare variable in private
(ii) Making setting and getting method for that variable.

For Example:

```
public class Encapsulation {
        private int data1 ;
        private int data2 ;
    public void setData1 ( int data1)
    {     this. data1 = data1 ;
    }
    public void setData2 (int data2)
    {   this. data2 = data2 ;
    }
   public int getData1 ()
     {   return data1 ;
     }
  public int getData2 ()
     {  return data2;
     }
}
```

(d)

Abstraction is the quality of dealing with ideas rather than events. For example when we consider the case of email, complex details such as what happens as soon as we send and email. In object oriented programming abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

(i) A class which contains the abstract keyword in its declaration is known as abstract class.

(ii) If a class is declared abstract it cannot be instantiated

(iii) Abstrac classes may or may not contain abstract methods.

(e)

Yes, abstract can contains main method in Java. The main method is a static method so it is associated with class, not with object/instance. The abstract is applicable to the object so there is no problem if it contains the main method. In main method we can not create an instance of the abstract class but we can instantiate other concrete class.

(f)

Abstraction can be achieve by two ways

(i) abstract class which will extends other class

(ii) Interface which will implement in class.

to use an abstract class, you have to inherit it from another class, provide implementations to the abstract method in it.

If we inherit an abstract class we have to provide implementations to all the abstract methods in it.

abstract method also can achive abstraction.

(9) A method which does not have body is known as abstract method. It contains only method signature with semi colon, an abstract keyword before it.

```
public abstract myMethod();
```

To use an abstract method we need to inherit it by extending its class and provide implementation to it. A class which contains 0 or more abstract is known as abstract class. If it contains at least one abstract method, it must be declared abstract. Once we extend an abstract class in Java we need to override all the abstraction method in its or declare it abstract. If we don't a compile time error will be generate for each abstract method saying subclass name is es not abstract and does not override abstract method abstract method-name in class name.

(h) The process by which one class acquires the properties and functionalities of another class is called inheritance. The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from the another class.
There have two class relationship ~~beet~~ in inheritance
This relationship also called IS-A relationship.

(i) Child class
(ii) Parent class

Child class: The class that extends the feature of another class is known as child class, sub class or derived class.

Parent Class: The class whose properties and functionalities are used (or inherited) by another class is known as parent class, super class or Base class.

~~To make relation~~
To make inherited between two class we need to use extends keyword

```
Subclass extends Superclass {
        Rest of the code
}
```

(1)

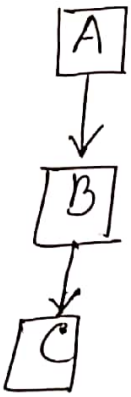
Extends: In Java, the extends keyword is used to indicate that the class which is being defined is derived from the base class using inheritance. So basically, extends keyword is used to extend the functionality of the parent class to the subclass. A class can extend only one class to avoid ambiguity.

Implements: In Java, the implements keyword is used to implement an interface. An interface is a special type of class which implements a complete abstraction and only contains abstract methods. To access the interface methods, the interface must be "implemented" by another class with the implements keyword and the methods need to implemented in the class which is inheriting the properties of the interface. Since an interface is not having the implements of the method, a class can implement any number of interfaces at a time.

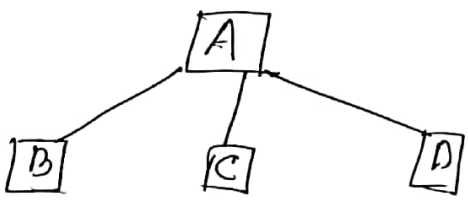
(i)

(I) Single Inheritance: When a class extends another one class only then we call it sing inheritance.

(II) Multilevel Inheritance: Multilevel inheritance refers to a mechanism in OO (object oriented) technology where one can inherit from a derived class, there by making this derived class the base class for the new class. As we can say c class is subclass of B class and B class is subclass of A class.



(III) Hienarchical Inheritance: When more than one classes inherit a same class then this is hienarchical inheritance. For Example class B, C, D extends same class A.



(k) ~~We conf~~

final keyword in java is used to restrict the user. We can use final Keyword in

(i) a variable
(ii) method
(iii) class.

We use final keyword for some advantage like

(i) stop value changing
(ii) stop Method Overriding
(iii) Stop Inheritance.

Final keyword is seems like a constant. Once declare we cant change it.

(1)

A final variable that is not initialized at the time of declaration is known as blank final variable. If we want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful. We can also initialize final variable but only in constructor.

For Example:

```
class Student {
    final String school_dress ;
                  ↓
        this is blank final variable

public Student ( )
{
    this.school_dress = school_dress ;
                        ↓
                initialize final variable
}

}
```

(m)

A method cannot be abstract and final. Because we cannot override final methods in Java. But incase of abstract we must override an abstract method to use it. Therefore, we cannot use abstract and final together before a method. If we do it forcefully we will get a compile time error.

(n)

Packages are used in Java in order to prevent naming conflicts, to control access to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

A package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management. Programmers can define their own packages to bundle group of classes/interfaces etc.

Q. Constructor chaining is the process of calling one constructor with respect to current object.
Constructor chaining can be done in two ways:

(i) Within same class: It can be done using this() keyword for constructors in same class.

(ii) From base class: by using super() keyword to call constructor from the base class.

Constructor chaining occurs through inheritance. A sub class constructor task is to call super class's constructor first. This ensure that creation of sub class's object starts with initialization of data number of the super class. There ~~could~~ could be any numbers of classes in inheritance chain. Every constructor calls up the chain till class at the top is reached.

(P)

No we can't have this and super in the same constructor. Because in java this() and super() statement should be the first statement.

Moreover before the derived class gets loaded into the main memory the base/super/parent class gets loaded into the memory for its execution / invocation of parent class constructor.

this() statement invokes the default constructor of the given class.

super() statement invokes the default constructor of the parent of the given inherited class.

(Q)

Yes, an abstract class in Java have a constructor. because

① The main purpose of the constructor is to initialize the newly created object. In abstract class, we have an instance variable, abstract methods, and non-abstract methods. We need to initialize the non-abstract methods and instance variables, therefore abstract classes have a constructor

(II) Also, even if we don't provide any constructor the compiler will add default constructor in an abstract class.

(III) An abstract class can be inherited by any number of sub-classes, thus functionality of constructor present in abstract class can be used by them.

(IV) The constructor inside the abstract class can only be called during constructor chaining.

Answer to the Question no: 3

(a)

Abstraction	Encapsulation
(I) Abstraction is the process on method of gaining the information	(I) While encapsulation is the process on method to contain the information
(II) Problems are solved at the design on interface level.	(II) Problems are solved at the implementation level.
(III) Abstraction is the method of hiding the unwanted information	(III) Encapsulation is a method to hide the data in a single entity on unit along with a method to protect information from outside
(IV) We can implement abstraction using abstract class and interfaces	(IV) Encapsulation can be implement ed using by access modifier.
(V) Implementation complexities are hidden using abstract classes and interfaces.	(V) Encapsulation, the data is hidden using methods of getters and setters method.
(VI) The objects that help to perform abstraction are encapsulated	(VI) The objects that result in encapsulation need not be abstracted.

(6)

Abstract Class	Interface
(I) Abstract class can have abstract and non-abstract method.	(I) Interface can have only abstract methods only.
(II) Abstract class doesn't support multiple inheritance	(II) Interface supports multiple inheritance.
(III) Abstract class can have final non-final, static and non-static variables.	(III) Interface has only static and final variables.
(IV) Abstract class can provide the implementation of interface.	(IV) Interface can't provide the implementation of abstract class.
(V) The abstract keyword is used to declare abstract class	(V) The interface keyword is used to declare interface.
(VI) An Abstract class can extend another Java class and implement multiple Java interfaces.	(VI) An interface can extend another Java interface only.
(VII) An abstract class can be extended using keyword "extends"	(VII) An interface can be implemented using implements.

Page: 24

(c)

Inheritance	Polymorphism
(I) Inheritance is one in which a new class is created (sub-class) that inherits the features from the already existing class (super-class)	(I) Polymorphism is that which can be defined in multiple forms.
(II) It is basically applied to classes.	(II) It is basically applied to functions or methods.
(III) Inheritance supports the concept of reusability and reduces code length in object oriented programming	(III) Polymorphism allows the object to decide which form of the function to implement at compile time (overloading) as well as run time (overriding)
(IV) Inheritance can be single, hybrid, multiple, hierarchical and multilevel inheritance but java don't support hybrid and multiple inheritance	(IV) It can be compile time polymorphism (overload) as well as run time polymorphism.
(V) It is used in pattern designing	(V) It also used in pattern designing.

(a)

Compile time Polymorphism	Run time Polymorphism
(I) In compile time polymorphism the call is resolved by the compiler	(I) In Run time Polymorphism the call is not resolve by the compiler.
(II) It is also known a static binding Early binding and overloading as well.	(II) Method overriding is the runtime polymorphism having same method with same parameters or signature but associated in different classes.
(III) Method overloading is the compile-time polymorphism where more than one methods share the same name with different parameters or signature and different type of return.	(III) It is also known as Dynamic binding, Late Binding and overriding as well.
(IV) It is achived by function overloading and operator overloading.	(IV) It is achived by virtual functions and pointers.
(V) It provides fast execution because the method that needs to be executed is known early at the compile time.	(V) It provids slow execution as compare to early bineing because the method that needs to be executed is known at runtime.

(e)

Inheritance	Composition
(I) In inheritance we define the class which we are inheriting (super class) and most importing it cannot be changed at runtime.	(I) In composition we only define a type which we want to use and which can hold its different implementation also it can change at runtime. Hence, composition is much more flexible than inheritance
(II) Here we can only extend one class, in other words more than one class can't be extended as java do not support multiple inheritance	(II) Whereas composition allows to use functionality from different class.
(III) In Inheritance we need parent class in order to test child class	(III) Composition allows to test the implementation of the classes we are using indepent of parent or child class.
(IV) Inheritance cannot extend final class.	(IV) Composition allows code reuse even from final classes.
(V) It is an is-a relationship	(V) It is an has-a relationship.

(5)

Inheritance	Polymorphism
(I) Inheritance is one in which a new class is created (subclass) that inherits the features from the already existing class (superclass).	(I) Polymorphism is that which can be defined in multiple forms.
(II) It is basically applied class	(II) It is basically applied to functions or methods.
(III) Inheritance supports the concept of reusability and reduces code length in Object-Oriented Programming.	(III) Polymorphism allows the object to decide which form of the function to implement at compile-time (overloading) as well as run-time (overriding)
(IV) Inheritance can be single, hierarchical and multilevel inheritance.	(IV) It can be compiled-time polymorphism (overload) as well as run-time polymorphism
(V) It is used in pattern designing	(V) It is also used in pattern designing.

(9)

Association	Aggregation	Composition
(I) Association relationship is denoted using an arrow	(I) Aggregation relationship is denoted using a straith line with an empty arrowhead at one end.	(I) Composition relationship is denoted using a straight line with a filled arrowhead at any one of the ends.
(II) Association can exist between two or more class.	(II) Aggregation is a part of an associatio -n relationship.	(II) The composition is a part of an association relationship
(III) There can be one-one, one-many, many-one and many-many associa -tion present between the association classes.	(III) Aggregation is comidered as a weak type of association	(III) The composition is considered as a strong type
(IV) object are linked with each other	(IV) Linked objects are not dependent upon the other object	(IV) Objects are highly dependent upon each

Page: 29

(h)

Abstract Class	Final class
(I) Uses the "abstract" key word	(I) Uses the "final" keyword.
(II) This helps to achive abstraction	(II) This helps to restrict other classes from accessing its properties and methods
(III) For later use, all the abstract methods should be overridden	(III) Overriding concept does not arise as final class cannot be inherited
(IV) A few methods can be implement-ed and a few cannot	(IV) All methods should have implements
(V) Can be inherited	(V) Cannot be inherited
(VI) Cannot be instantiated	(VI) Can be instantiated

(1)

Errors	Exceptions
(I) Recovering from Error is not possible	(I) We can recover from exceptions by either using try-catch block or throwing exceptions back to caller.
(II) All errors in Java are unchecked type.	(II) Exceptions include both checked as well as unchecked type.
(III) Errors are mostly caused by the environment in which program is running	(III) Program itself is responsible for causing exceptions.
(IV) Error occur at runtime and not known to the compiler	(IV) All exceptions occurs at runtime but checked exceptions are known to compiler while unchecked are not
(V) They are defined in java.lang.Error package	(V) They are defined in Java.lang.Exception package.

(1)

checked Exception	Unchecked Exception
(I) Checked exceptions occur at compile time.	(I) Unchecked exceptions occurs at runtime.
(II) The compiler checks a checked exception.	(II) The compiler does not check these types of exception
(III) These types of exceptions can be handled at the time of compilation	(III) These types of exceptions cannot be a catch on handle at the time of compilation because they get generated by the mistakes in the program.
(IV) They are the sub-class of the exception class	(IV) They are runtime exception and hence are not a part of the Exception class.
(V) Here, the JVM needs the exception to catch and handle	(V) Here, the JVM not require the exception to catch and handle.

Page: 32

(K)

Throw	Throws
(I) This keyword is used for explicity throwing an exception.	(I) This key word is used for declaring any exception
(II) Programmer cannot dissminate checked exceptions using the throw keyword.	(II) Programmers can dissminated checked exceptions using throws keyword.
(III) An instance trail the throw keyword	(III) A class trails the throws keyword.
(IV) Many exceptions cannot be thrown	(IV) Many exception can be declared using the throws.
(V) We have to use the throw keyword inside in any method	(V) We have to use the throws keyword ~~inside~~ with any sign of the method.

IS-A Relationship	HAS-A Relationship
(I) It is known as an inheritance	(I) It is known as composition and Aggregation.
(II) The main advantage of IS-A relationship is code reuseablity.	(II) Also code reuseablity but don't need to have Is-a relation-ship.
(III) By using extends keyword we can make IS-A relation ship	(III) There is no specific key word to implement HAS-A relationship
(IV) ~~Inhe~~ IS-A relationship is static binding	(IV) HAS-A relationship is dynamic binding
(V) IS-A relation_ship is totally automatically class functionality.	(V) Has-A relationship is a part class functionality.

Answer to the Question no: 4

Figure : 1

Association in Java is a connection or relation between two separate classes that are set up through their Object. Association relationship indicates how objects know each other and how they are using each other's functionality. It can be one-to-one, one-to-many, many-to-one and many-to-many.

There are two forms of Association
Composition and Aggregation are the two special forms of association. Let's explain

Composition :

It is a "belongs-to" type of association. It simply means that one of the objects is a logically larger structure, it is often called a "has-a" relationship.

Aggregation :

Aggregation is also "has-a" relationship, but, what distinguishes it from composition, is that the life cycles of the objects are not tied. Both the entities can survive individually which means ending one entity will not affect the other entity. The main Reason ~~we need~~ to use Aggregation is to maintain code reusability.

Figure 2:

The following figure showing package in Java. A package as the name suggests is a pack of classes, interfaces and other package. In java we use package to organize our classes and ~~interit~~ interfaces. We have two types of packages in Java. built-in packages and the packages we can create. In this package two type of package.

(I) User defined class

(II) Built-in class.

in this figure here shown built in class which is

(I) java.lang

(II) java.io

(III) java.awt

(IV) java.math etc

different type of package have different type of code and sub packages like java.io has input output like file system. Java.util package ~~have~~ have Scanner subclass where are many forms of scanners.