**OBJECT OREINTED PROGRAMMING LANGUAGE**

**INTRODUCTION:**

* Identifying the state and behaviour real – world objects is a great way to begin thinking in terms of object oriented programming.
* Objects are physical world entity. Whenever you encounter an object in real life ask yourself two questions.
* What states can that object have?
* What behaviour can that object exhibit OOPS as the name suggests uses objects in programming?

**ELEMENTS OF OOPS:**

* The four principles of objected oriented programming are
* Abstraction.
* Encapsulation.
* Polymorphism
* In heritance.
* Let us discuss in detail.

**ABSTRACTION:**

* It is a process where you show only “relevant” data , “hide” unnecessary details of an object from the user.
* For example:
* When you login your amazon account online
* You enter your user-id and password and press login, what happens when you press login ,how the input data sent to amazon server, how it gets verified is all abstracted from you.
* You only show what’s important to the user and hide the background complexity details.

**ENCAPSULATION:**

* It has a few logically diff object which communicate with each other according to the rules defined in the program.
* Encapsulation is achieved when each object keeps its state private, inside a class. Other objects don’t have direct access to this state.
* Instead, they can only call a list of public functions called methods.
* So, the object manages its own state via methods and no other class can touch it unless explicitly allowed.
* If you want to communicate with the object, you should use the methods provided. But (by default) you can’t change the state.

**POLYMORPHISM:**

* The condition occurring in several different forms. They are:
* A son to your father.
* A student to your sister.
* A buddy to your friend.
* A brother to your sister.
* A single person exhibit many roles in life, this is known as polymorphism.
* Polymorphism means to process objects differently based on their data type.
* In other words it means, one method with multiple implementations.

**INHERITANCE:**

* It is a mechanism in which one class acquires the property of another class.
* For example: a child inherited, the traits of his/her parents.
* The generic traits of a parent is exhibited by its children and goes on as a chain.
* With inheritance, we can reuse the fields and methods of the existing class.
* Hence, inheritance facilities reusability and is an important concept of oops.

**CLASS:**

* A class represents a collection of having the same characteristic properties that exhibit common behaviour.
* It gives the blueprint destination of the objects that can be created from it.
* Creation of an object as a member of a class is called instantiation.
* Thus object is an instance of a class.

**QUESTIONS:**

* Which feature of oops indicate code reusability?

1. Encapsulation 3. inheritance
2. Abstraction 4. polymorphism

**Answer:** option 3

* In a function can perform more than 1 type of tasks, where the function name remains same, which feature oops is used here?

1. Encapsulation 3.Inheritance
2. Abstraction 4.polymorphism

Answer: option 4

**COMPARISION WITH PROCEDURAL PROGRAMMING:**

* Object oriented design started the moment computers were invented.
* Programming is basically giving certain instructions to the computer.
* Object contain data, in the form of fields (often known as attributes) and code in the form of procedures (often known as methods).
* Programs are made up of modules.
* They are part of program that can be coded and tested separately.
* These parts can be assembled to form a complete program.
* Pop is a top-down approach.
* Programs are divided into small chunks based on the function.
* Each function contains different data.
* It follows a systematic approach to solve a problem.
* Functions are more important than data in a program.
* Different parts of a program are interconnected via parameter passing.
* Oops takes a bottom approach in designing a program.
* Program is divided in to objects on the problem.
* Each object controls its own data.
* Focuses on the security of the data irrespective of the algorithm.
* The main priority is data rather than functions in a program.
* The functions of the objects are linked via message passing.
* Objects are collections of information that are treated as a singular entity.
* Classes are sort of like pre-objects.
* They contain a list of attributes that which defined became an object.
* An object simply defines one particular instance of a thing belonging to that class.
* So we could have an object called white queen.
* That object would have definitions for all four attributes (white, tall, cylindrical any number of spaces in a direction).
* It might have methods or functions.
* So far we got the difference between class and object.

**QUESTIONS:**

* Which of the following best define a class?

1. Parent of an object 3. Instance of an object
2. Blueprint of an object 4. Scope of an object

Answer: option 2

* Pure oops can be implemented without using class in a program

1. True 2. False

Answer: option 2

**MERITES AND DEMERITES OF OOPS:**

* Oops technique is the latest programming approach.
* The primary factor is to remove the major disadvantages of the pop approach.
* Although oops has proved revolutionary in the world of software development yet it have some disadvantages too.
* But if we compare both the oops and pop, the ratio of pros are very high as compared to the cones.

**ADVANTAGES OF OOPS:**

* The programs written with oops are really easy to understand. We can model a real-world concept using oops
* Oops approach offers the reusability of classes.
* We can reuse the classes that are already created without writing them again and again.
* Since the parallel development of classes if possible in oops concept, it result in the quick development of complete programmes.
* Programmes written in oops technique are marginally easier to test, manage as well as maintain.
* It is a secure development technique since data is hidden and can’t be accessed by external functions.

**DISADVANTAGES:**

* Sometimes, the relation among the classes becomes artificial in nature
* Designing a program in oops concept is a little bit tricky.
* The program should have proper planning before designing a program using oops approach.
* The programmers need proper skills such as design skills, program skills, thinking in terms of objects etc.
* The size of programs developed with oops is larger than the procedural approach.
* Since larger in size means more instruction to be executed, which results in the slower execution of programmes.

**SUMMARY:**

* According to you what is the object?

1. Real world entity
2. Any physical object with state
3. Child of a class
4. Overview of a class

Answer: option 1 and 2

* Which of the following is not a feature of oops?

1. Classes must be used
2. In heritance
3. Data may or may not be declared using object.
4. Function overloading

Answer: option 3

**CONCEPT OF AN OBJECT:**

* The state of a object consist of data fields with their current values.
* The state represents the cumulative results of an object behaviour.
* A property is an inherent characteristic, trait, or a quality that distinguishes one kind of object from another.
* The properties in a class are usually static.
* All properties have some value.
* May be a simple quantity.
* Behaviour is how an object acts and reacts in terms of its state changes and message passing.
* When you send a message to an object, you actually invoke a method.
* Invoking a method will cause certain well defined behaviour, and may change the objects current state.
* A few kinds of operations that a client may perform an object:
* Modifier: alerts the state of an object.
* Selector: access the state of an object.
* Constructor: creates an object and initialize

the state.

* Destructor: destroy an object.
* An object has both state and behaviour.
* The state defines the object whereas, the behaviour defines what the object does.
* Objects are real time entities which are created through their template their class.
* By real time entity, we mean they would occupy a memory space.
* These values create an object.
* Since object occupies a space in memory so they have an unique address in memory.
* This becomes an identity of an object.

**QUESTION:**

* Attributes are assigned value

1. When operations are performed to an object.
2. When instance of objects are defined.
3. When methods are involved.
4. When classes are identified.

Answer: option 2

**CLASSES AND ACCESS MODIFIERS:**

* A class represents a collection of objects having the same characteristic properties that exhibit common behaviour.
* It gives the blueprint the objects that can be created from it.
* Creation of an object as a member of class is called instantiation of class. thus, object is instance of a class.
* A class is a blueprint for an functional entity which defines its properties and its functions.
* It perform various actions.
* General syntax of the classes:

Class classes name {

Access \_ specifier;

Member \_ variables;

Member \_ functions;

} object1, object2

* We can define a class using the class keyword and the class body enclosed by a pair of curly braces.
* For example:

Public class humanbeing {

//declare field properties, event, delegate, and method

}

**QUESTION:**

* Which among the following the main characteristic of class?

1. Inheritance 3. Encapsulation
2. Polymorphism 4. Object creation

Answer: option 1

* Access modifiers are keyword in object oriented languages.
* These set the accessibility of classes, methods, and other members.
* Access modifiers are a specific part of programming language syntax, it is used to facilitate the encapsulation of components.
* So an access specifier tasks a certain class, method, or variable and decide what other classes are allowed to use them.
* Access specifiers are used to protecting data from issues.

**PUBLIC CLASS MEMBERS:**

* Public class members and functions can be used from outside of a class by any function or other classes.
* All the class members declared under public will be available to everyone.
* The public members of a class can be accessed from anywhere in the program.
* This is done with the help of the object of that class.

**PROTECTED CLASS MEMBERS:**

* Protected class members and functions can be used in the class.
* Protected members and functions cannot be accessed from other classes directly.
* Protected data members and functions can be used by the class derived from this class.

**PRIVATE ACCESS SPECIFIER:**

* All class members are private by default.
* We can access those members only with in the class.
* We cannot access from outside of the class.
* But there is a possibility to access the private class members from outside class is possible only through the methods which are declared in public.
* It provides security to the class members.
* Usually the data members are written in private functions.
* A bit part of oops programming is encapsulation.
* Access specifier allows encapsulation.
* Encapsulation can make debugging a IOT easier. What a outside world cannot see it cannot depend on!
* The object is a ‘firewall’ between the object and the outside world.
* The hidden data and methods can be changed without affecting the outside world.

**SUMMERY:**

* If the members have to be accessed from anywhere in the program and other packages also, which access specifier should be used?

1. Public 3.Protected
2. Private 4.Default

Answer: option 1

* Which access specifier is/are most secure during inheritance?

1. Public 3.Protected
2. Default 4.Private and default

Answer: option 3

* Which among the following best describes the protected specifier?

1. Members are most secure and can’t be used outside the class.
2. Members are secure but can be used outside the class.
3. Members are secure as private, but can be inherited.
4. Members are secure as private, but can’t be inherited.

Answer: option 3

**EXPLORING CLASSES:**

**CLASS DIAGRAMS- UML**

* Class diagram is a static diagram.
* It represents the static view of an application.
* Class diagram is not only used for visualizing, describing and different aspects of a system but also for constructing executable.
* Class diagram describes the attributes and operation of a class.
* It also defines the constraints imposed on the system.
* The class diagrams are widely used in the modelling of oops.

**PURPOSE OF CLASS DIAGRAM:**

* The purpose of the diagram is to model the static view of an application. Class diagrams are only diagrams which can be directly mapped with oops.
* UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application however class diagram is a bit different.
* It is most popular UML diagram in the coder community.
* The purpose of class diagram can be summarized as:
* Describe the responsibility of a system.
* Base of component and development diagrams.
* Forward and reverse engineering.
* The following points should be remembered while drawing a class diagram.
* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified in advance.
* Responsibility of each class should be clearly identified.
* For each class, minimum number of properties should be specified as unnecessary properties will make the diagram complicated.
* Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.
* In a nutshell it can be said ,class diagrams are used for:
* Describing the static view of the system.
* Showing the collaboration among the elements of static view.
* Describing the functionalities performed by the system.
* Construction of software applications using oops.

**MESSAGE PASSING AND STATIC MEMBERS:**

* Interaction between object happens by messages being sent.
* A message activates a method on the calling object.
* An object O1 interacts with another object O2 “O1 sends O2 a message”.
* O1 and O2 must be related to communication.
* The call of a method corresponds to a function call in a non- object- oriented language such as c or pascal.
* The process of programming in an oops therefore involves the following basic steps:
* Creating classes that define objects and their behaviour.
* Creating objects from class definitions.
* Establishing communication among objects.
* The members of a class can be declared using the storage class modifier static.
* These data members are shared by all instances of this class and are stored in one place.
* Non-static data members are created for each class object variable.
* The inability to declare static members of a class would have led to the need to declare these data on the global level of the program.
* It would break the relationship between data and their class.
* It is not consistent with the basic paradigm of the oops.
* The static member allows class data that are not specific to a particular instance to exist in the class scope.
* When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of a static member.
* A static member is shared by all objects of the class.
* All static data is initialized to zero when the first object is created.

**STATIC FUNCTION MEMBERS:**

* By declaring a function member as static, you make it independent of any particular object of the class.
* A static member function can be called even if no objects of the class exist.
* Also the static functions are accessed using only the class name and the scope resolution operator ”::”
* A static member function can only access static data member, other static member function and any other functions from outside the class.
* Static member functions have a class scope and they do not have access to this pointer of the class.
* You could use a static member function to determine whether some objects of the class have been created or not.

**QUESTION:**

* The static member functions?

1. Have access to all the members of a class.
2. Have access to only constant members of a class.
3. Have access to only the static members of a class.
4. Have direct access to all other class members also.

Answer: option 3

* We can use the static member functions and static data member?

1. Even if class object is not created.
2. Even if class is not defined.
3. Even if class doesn’t contain any static member.
4. Even if a class doesn’t have complete definition.

Answer: option 1

**CONSTRUCTION AND DESTRUCTION OF OBJECTS**

**CONSTRUCTORS:**

* Constructors are special class functions which perform initialization of every object. The compiler calls the constructor whenever the object is created.
* Constructors initialize values to object members after storage is allocated to the object.
* Let’s start the constructions first, above mentioned is the syntax of defining a constructor function in a class.
* Remember that the name of the constructor will be same as the name of the class.
* Constructors will never have a return type.
* Constructors are of three types:
* Default constructor.
* Parameterized constructor.
* Copy constructor.

**DEFAULT CONSTRUCTORS:**

* Default constructor is a constructor which doesn’t take any argument.
* It has no parameters.
* Syntax:

Class \_ name()

{

//constructor definition

}

* A default constructor is so important for initialization of object members, that even if we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.

**PARAMETERIZED CONSTRUCTORS:**

* These are the constructors with parameters.
* Using this constructor you can provide different values to different members of different objects, by passing the appropriate values as argument.
* Syntax:

Class \_ name (parameter1, parameter2, …………..)

{

//constructor definition

}

**COPY CONSTRUCTOR:**

* Copy constructor is a type of constructor which is used to create a copy of an already existing object of a class type.
* It is usually of the form X(X&), where X is the class name.
* The compiler provides a default copy constructor to all the classes.
* Syntax:

Class name(const class name and object name)

{

…………….

}

* As it is used to create an object, hence it is called a constructor. And, it creates a new object, which is an exact copy, hence it is called copy constructor.

**DESTRUCTOR:**

* Destructor is a special class function which destroys the object as soon as the scope of object ends.
* The destructor is called automatically by the compiler when the object goes out of the scope.
* The syntax for destructor is same as that for constructor.
* The class name is used for the name of destructor, with a tilde ~ sign as prefix to it.
* Destructors will never have any arguments.

**SUMMERY:**

* Which among the following is called first, automatically whenever the object created?

1. Class 3.New
2. Constructor 4.Trigger

Answer: option 2

* How many types of constructors are available for using general (with respect to parameters)?

1.2 2.3 3. 4 4. 5

Answer: option 1

**LET’S EXPLORE RELATIONSHIPS**

**INHERITANCE: PURPOSE AND ITS:**

* Inheritance is the process of creating a new class, called the derived class, from the existing class, called the base class.
* The inheritance has many advantages, the most important of them being the reusability of code.
* Rather than developing new objects from scratch, new code can be based on the work of other than developers, adding only the new features that are needed.
* The reuse of existing classes saves time and effort
* However, inheritance may be implemented in oops as illustrated in figure and they include:
* Single inheritance.
* Multi level inheritance.
* Hierarchical inheritance.
* Hybrid inheritance.
* Multipath inheritance.
* Multiple inheritances.

**SINGLE INHERITANCE:**

|  |
| --- |
| Base class |



|  |  |
| --- | --- |
| Derived class |  |
|  |  |

* When a derived class to inherit properties and behaviour from a single base class, it is called single inheritance.

**MULTILEVEL INHERITANCE:**

* A derived class is created from another derived class is called multilevel inheritance.

**HIERARCHICAL INHERITANCE:**

* More than one derived class are created from single base class, is called hierarchical inheritance.

**HYBRID INHERITANCE:**

* Any combination of the above inheritance (single, hierarchical and multilevel) is called hybrid inheritance.

**MULTIPLE INHERITANCES:**

* Multiple inheritances allow programmers to create classes that combine aspects of multiple classes and their corresponding hierarchies.

**QUESTION:**

* Multiple inheritances is when a class is derived from?

1. Another class 3.The other two derived class.
2. Two or more classes 4. Exactly one class

Answer: option 2

**HOW ‘IS A RELATIONSHIP’ WORKS**

**‘HAS A ASSOCIATION’:**

* Arrange going left to right says “a college has zero or more students”.
* Arrow going right to left says “a student has exactly 1 college”. What it means in real terms is that the college class will contain a variable that somehow links to set a student will have a variable that references a college object.
* Note that they are only linking classes: we don’t start drawing arrows to primitive types.
* The graphical notation used here is part of UML( unified modelling language).
* UML is basically standardised set of diagrams. It can be used to describe software independently of any programming language used to create it.
* The line between them shows the “is - a” relationship.
* The arrow points to the parent class from the child class. It can be read as” a ford is – an automobile”.
* The phrase” is - a” is in common use in computer science. The arrow between a child and parent is sometimes called an “is – a link”. The clouds represent classes. Inheritance between classes, not between objects.
* A parent class is a blue print that is followed when an object is constructed.

**ASSOCIATION AND AGGREGATION- ‘HAS A RELATIONSHIP’:**

* Association is a “has - a” type relationship. It establish the relationship between two classes using through their objects. It can be one to one, one to many, many to one and many to many.

**AGGREGATION:**

* Aggregation is based on” has - a” relationship. It is a special form of association. In association there is not any classes work as owner but in aggregation one entity work as owner.
* In aggregation both entities meet for some work and then get separated. Aggregation is a one way association.
* Let us take an example of “student” and “address”. Each student must have an address so relationship b/w student class and address class will be “has – a” type relationship. But vice versa is not true. So student work as owner entity. This will be aggregation relationship.
* Idea: make new objects by combining existing objects.
* Reusing the implementation.
* An aggregation consist of a number of (sub-) concepts which collectively is considered a new concept.
* A decomposition splits a single concept in to a number of (sub-) concept.

**SUMMERY:**

* Which among the following best describes the inheritance?

1. Copying the code already written.
2. Using the code already written once.
3. Using already defined functions in programming language.
4. Using the data and functions into derived segment.

Answer: option 4

* If hierarchical inheritance requires to inherit more than one class to single class, which syntax is correct? (a, b, c are class names) ?

1. Hierarchical class a: public b, public c
2. Multiple class a: public b, public c
3. Many class a: public b, public c
4. class a: public b, public c

Answer: option 4

* If object of lowest level class is created( last derived class),\_\_\_\_\_\_\_\_\_\_\_\_\_ of its parent class constructors are called

1. Few 3.Only parent and parent
2. All 4.Base and derived.

Answer: option 3

**INTERFACES AND ABSTRACT CLASS**

**LET’S EXPLORE INTERFACES:**

* An interface is a description of the actions that an object can do.
* In oops, an interface is a description of all functions that an object must have to be an order “x”.
* Anything that “acts like” a light, should have a turn\_ on () method and a turn\_ off () method.
* The purpose of interfaces is to allow the computer to enforce these properties. And to know that an object of TYPE T (whatever the interface is) must have functions called X, Y, Z, etc.
* An interface has a very simple syntax that looks very much like a class definition.
* Interfaces are placed in their own files which have the same name as the interface (are capitalized) and end with a familiar language extension (e.g. “as”).
* To tell the computer that a new class that you are waiting will fulfil all the requirements of an interface. You must use the keyword implements in the same location where you can use the keyword extends.

**QUESTION:**

* Which of these keywords is used by a class to use an interface defined previously?

1. import 3.implements
2. Import 4.Implement

Answer: option 3

* Which of the following is the correct way of implementing an interface salary by class manager?

1. Class manager extends salary {}
2. Class manager implements salary {}
3. Class manager imports salary {}
4. None of the mentioned.

Answer: option 2

**ABSTRACT CLASSES:**

* Abstraction is one of the key concepts of oops language.
* It’s main goal is to handle complexity by hiding unnecessary details from the user. That’s a very generic concept that’s not limited to oops. You can find it everywhere in the real world.
* I’m a coffee addict. so, when I wake up in the morning, I go into my kitchen, switch on the coffee machine and make coffee. Making coffee with a coffee machine is a good example for abstraction.
* You need to know how to use your coffee machine to make coffee. You need to provide water and coffee beans, switch it on and select the kind of coffee you want to get. The thing you don’t need to know is how the coffee machine is working internally to brew a fresh cup of delicious coffee.
* You just interact with simple interface that doesn’t require any knowledge about the internal implementation.

**ABSTRACTION IN OOPS:**

* Objects in oops language provide an abstraction that hides the internal implementation details. You just need to know which methods of the object are available to call. Which input parameters are needed to trigger a specific operation.
* But you don’t need to understand how this method is implemented. Which kinds of actions it has to perform to create the expected result.
* Abstract class is nothing special to look at when you want to differentiate with a regular class. But it got some new methods which are unimplemented, or only declared, not defined.
* In this case, all those methods and the class itself are renamed with the “abstract” keyword. An abstract class can have one or more multiple number of unimplemented methods.
* As an abstract class is not proper, or does not have ideally defined methods compared to a regular class. So, abstract classes cannot be instantiated, that means we create any direct objects of abstract classes.
* The classes which inherit/extend this abstract class, have to implement/define all the abstract methods of the parent abstract class.
* If any class fails to define any of the abstract methods, that class also becomes an abstract class.

**ABSTRACT METHODS AND CLASS:**

* Abstract methods are mostly declared where two or more subclasses are also doing the same thing in different ways.
* It also extends the same abstract class and offers different implementations of the abstract methods. Abstract classes help to describe generic types of behaviours and oops class hierarchy.
* It also describes subclasses to offer implementation details of the abstract class.

**QUESTION:**

* Which among the following best defines the abstract methods?

1. Functions declared and defined in base class.
2. Functions only declared in base class.
3. Functions which may or may not be defined in base class.
4. Functions which must be declared in derived class.

Answer: option2

**SUMMERY:**

* Which of these can be used to fully abstract a class from its implementation?

1. Objects 3.Interfaces
2. Packages 4. None of them

Answer: option 3

* Which of these access specifiers can be used for an interface?

1. Public 3.Private
2. Protected 4.All of them

Answer: option 1

* Which among the following is true?

1. Abstract methods can be static.
2. Abstract methods can be defined in defined in derived class.
3. Abstract methods must not be static.
4. Abstract methods can be made static in derived class.

Answer: option 3

**CONCEPT OF POLYMORPHISM**

**EARLY BINDING AND LATE BINDING:**

* The compiler performs a binding a process called binding when an object is assigned to an object variable.
* There are two types of binding:

1. The early binding (static binding) refers to compile time binding.
2. The late binding (dynamic binding) refers to runtime binding.

* When performing early binding, an object is assigned to a variable declared to be of a specific object type.
* Early binding objects are basically a strong type objects or static type objects
* The biggest advantage of using early binding is for performance and ease of development.
* While performing early binding the compiler can ensure at compile time that the function will exist and be callable at runtime.
* More over the compiler guarantees that the function takes the exact number of arguments.
* Also that they are of the right type and can check that the return value is of correct type.
* By contrast, in late biding functions, methods, variables and properties are detected and checked only at the run time. It implies that the compiler is unaware what kind of object or which methods or properties an object contains until run time.
* The biggest advantage of late binding is that the objects of this type can hold references to any object. But they lack many of the advantages of early- bound objects.
* While performing late binding there is a possibility of the target function may not exist. While perform late binding there is a possibility of the target function may not exist.
* Also the target function may not accept the arguments passed to it. It may have a return value of the wrong type.
* Late binding the target method is looked up at run time.
* Most script languages use late binding, and compiled languages use early binding.

**QUESTION:**

* In which of the following mechanisms, types of all variables and expressions are fixed at compilation time?

1. Strong typing.
2. Weak typing.
3. Static binding/ early binding.
4. Dynamic binding/ late binding.

Answer: option 3

**METHOD OVERRIDING AND OVERLOADING:**

* Polymorphism is another oops concept. It means the ability to take more than one form. An operation may exhibit different behaviours at different instances.
* The instances depend upon the type of data used in the operation. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
* Real life example of a polymorphism, a person at the same time can have the same characteristics.
* Polymorphism are divided into two types.

1. Compile time polymorphism
2. Run time polymorphism

* Method overloading happens at compile time (early binding). While overriding happens at run time (late binding).
* In method overloading, method call to its definition has happen at compile time (static binding). While in method overriding, call to its definition happens at runtime (dynamic binding).

**FUNCTION OVERLOADING:**

* In function overloading we can see the function names are same but parameter type and number of parameters are different.
* When two or more methods (functions) in the same class have the same name but different parameters is called method overloading. This means that we can use the same function name to create functions that perform a variety at different tasks. This is also known as polymorphism.
* Using the concepts of functions overloading, we can design a family of functions with one function name but with different argument list.

**OPERATOR OVERLOADING:**

* The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.
* For example, we can make the operator (‘+’) for string to concatenate two strings. We know that this is the additional operator whose task is to add two operands.
* So in a single operator ‘+’ when placed between two operands, adds them and when placed between strings operand, concatenate them.
* Basically, operator overloading is used because it make programs more readable as the operator which are used for basic data types can also be used for user- defined data types.
* Almost any operator can be overloaded in c++. However there are few operators which cannot be overloaded. Operator that are not overloaded are follows:

1. Scope operator -::
2. Size of
3. Member selector -.
4. Member pointer selector -\*
5. Ternary operator -?:

**METHOD OVERRIDING:**

* When two or more methods (functions) have the exact same method name, return type, number of parameters, and types of parameters as the method in the parent class is called method overriding.
* Here are the points to be taken care of:
* The argument list should be exactly the same as that of the overridden method.
* The return type should be the same or a subtype of the return type declared in the original overridden method in the super class.
* Instance methods can be overridden only if they are inherited by the subclass.
* A method declared final cannot be overridden.
* A method declared static be inherited cannot be overridden but can be declared.
* If a method cannot be inherited, then it cannot be overridden.
* A subclass with in a same package as the instances super class can override the any super class method that is declared or not.
* A subclass in a different package can only override the non- final methods declared public protected.

**SUMMERY:**

* Exactly same declaration in base and derived class includes?

1. Only same name.
2. Only same return type and name.
3. Only the same return type and argument list.
4. All the same return type, name and parameter list.

Answer: option 4

* How to access the overridden method of the base class from the derived class?

1. Using arrow operator.
2. Using dot operator.
3. Using scope resolution operator.
4. Can’t be accessed once overridden.

Answer: option 3

* The functions to be overridden?

1. Must be private in base class.
2. Must not be private base class.
3. Must be private in both derived and base class.
4. Must not be private in both derived and base class.

Answer: option 2

**EXCEPTION HANDLING**

**DIFFERENT TYPES OF HANDLING:**

* Exceptions are run- time compilers. They are abnormal conditions that a program encounters during its execution.
* There are two types of exceptions:

1. Synchronous.
2. Asynchronous (ex: which are beyond the program’s control, disc failure etc).

* Oops provides following specialized keywords for this purpose.
* Try: Represents a block of code that can throw an exception.
* Catch: Represents a block of code that is executed when a particular exception is thrown.
* Throw: Used to throw an exception also used to list the exceptions that a function throws, but doesn’t handle itself.
* Following are the main advantages of exception handling over traditional handling:

1. Separation of error handling code from normal code:

* In traditional error handling codes, there are always if else conditions to handle errors.
* With try catch blocks, the code for error handling becomes separate from the normal flow.

1. Functions/methods can handle any exceptions they choose:

* A function can throw many exceptions, but may choose to handle some of them.

1. Grouping of error types:

* Both basic types and objects can throw the exception.
* Exceptions can be categorized into two types:
* Compile time exceptions.
* Run time exceptions.
* In a program is syntactically correct- that is, free of syntax errors
* It will run by the interpreter.
* However, the program may exit unexpectedly during execution if it encounters a runtime error. They are a problem which was not detected when a program was parsed. It is only revealed when a particular line is executed.
* When a program comes to a halt because of a runtime error, we say that is crashed. Consider the English instruction **flap your arms and fly to Australia.**
* While the instruction is structurally correct and you understand its meaning. Perfectly, it is impossible for you to follow it.
* Some examples of runtime errors:
* Division of zero.
* Performing an operation an in compatible types.
* Using an identifier which has not been defined.
* Accessing a list element, dictionary value or object attribute which doesn’t exist.
* Trying to access a file which doesn’t exist.

**CHECKED EXPECTION:**

* Checked exceptions are also known as compile- time exceptions.
* These exceptions are checked by the compiler during the compilation process to confirm weather the exception is handled by the programmer or not. If not, then the system displays a compilation error.
* For example, **SQL Exception, IO Exception, Invocation target exception and Class not found exception.**

**QUESTION:**

* What is an exception?

1. Problem arising during compile time.
2. Problem arising during run time.
3. Problem in syntax.
4. Problem in ide.

Answer: option 2

* An exception may arise when?

1. Input is fixed.
2. Input is some constant value of program.
3. Input given is invalid.
4. Input is valid.

Answer: option 3

**CREATING USER-DEFINED EXCEPTION:**

* We have already defined, exception classes such as arithmetic exception, null pointer exception etc. These exceptions are already set to trigger on predefined conditions such as when you divide a number by zero it triggers arithmetic exceptions.
* In java we can create our own exception class and throw that exception using throw keyword. These exceptions are known as user- defined or custom exceptions.
* In this topic we will see how to create your own custom exception and throw it on a particular condition.
* Let’s look an example code:



* Output for this code:

Starting of try block

Catch block

My exception occurred: this is my error

Message

**EXPLANATION:**

* You can see that while throwing custom exceptions.
* I gave a string in parenthesis:

Throw new my exception (“this is my error message”) ;)

* That’s why we have a parameterized constructor (with string parameter) in my custom exception class.
* Note the following points:
* User- defined exception must extend exception class.
* The exception is thrown using the throw keyword

**SUMMERY:**

* If a file that needs to be opened is not found in the target location then

1. Exception will be produced.
2. Exceptions are not produced.
3. Exception might get produced because of syntax.
4. Exceptions are not produced because of logic.

Answer: option 1

* Which are the two blocks that are used to check error and handle the error?

1. Try and catch 3. Do and while
2. Trying and catching 4. Try do and check

Answer: option 1

**INTRODUCTION TO MULTITHREADING:**

* Multithreading in oops is a process of executing multiple threads simultaneously. A thread is a lightweight sub- process, the smallest unit of processing. Multiprocessing and multithreading are used to achieve multitasking.
* However, we use multithreading than multiprocessing because threads use a shaded memory area. They don’t allocate separate memory area to save memory. Also context switching is b/w the threads take less time than process.

**ADVANTAGS OF MULTITHREADING:**

* It doesn’t block the user because the threads are independent and you can perform multiple operations at the same time.
* You can perform many operations together, so it saves time.
* Threads are independent, so it doesn’t affect other threads if an exception occurs in a single thread.
* A thread is a light weight process, the smallest unit of processing. It is a separate path of execution. Threads are independent. If there occurs an exception in one thread, it doesn’t affect other threads.
* It uses a shaded memory area. A thread is executed inside the process. There is a context switching b/w the threads.
* There can be multiple process inside the OS, and one process can have multiple threads.
* They are two ways to create a thread:

1. By extending thread class.
2. By implementing the runnable interface.

**THREAD CLASS:**

* Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends object class and implements the runnable interface.
* Commonly used constructors of thread class:
* Thread ()
* Thread (string name)
* Thread (runnable r)
* Thread (runnable r, string name)
* A thread can be in one of the five states.
* New
* Runnable
* Running
* Non- runnable (blocked)
* terminated
* The thread state are as follows:
* New: the thread is a new state if you create an instance of thread class but before the invocation of the start () method.
* Runnable: the thread is an runnable state after invocation of start () method, but a thread schedular has not selected it to be the running thread.
* Running: the thread is in running state if the thread schedular has selected it.
* Non- runnable: this is the state when the thread is still alive, but is currently not eligible to run.
* Terminated: a thread is terminated or dead state when its run () method exists.

**QUESTION:**

* What is multithreading?

1. It is a process in which two different processes run simultaneously.
2. It’s a process in which two or more parts of the same process run simultaneously.
3. It’s a process in which many different parts are able to access the same information.
4. It’s a process in which a single process can access information from many sources.

Answer: option 2

**GARBAGE COLLECTION:**

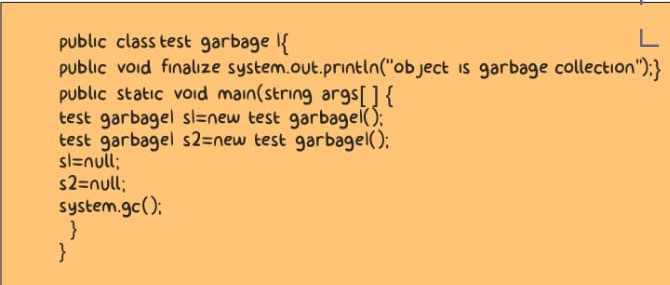
* Garbage means unreferenced objects. Garbage collection is the process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.
* To do so, so we were using free () function in the c language and delete () in c++. But in java it is performed automatically. So, java provide better memory management.
* Garbage collection is essentially the opposite of manual memory management. It required the programmer to specify which objects to deallocate and return to the memory system. Like other memory management techniques, garbage collection may take a significant proportion of total processing time in a program.
* With good implementation and with enough memory, depending on application, garbage collection can be faster than manual memory management.

**ADVANTAGES:**

* Garbage collection frees the programmer from manually dealing with memory deallocation.
* It makes the memory efficient because garbage collector removes the unreferenced objects from heap memory.
* It is automatically done by the garbage collector so we don’t need to make extra efforts.

**UNREACHABLE OBJECTS:**

* An object is said to be unreachable if it doesn’t contain any reference to it. Also note that objects which are part of the island of isolation are also unreachable. An object is said to be eligible for GC (garbage collection) if it is unreachable. After I =null; integer object 4 in heap area is eligible for garbage collection.



* Output:

Object is garbage collected

Object is garbage collected

**QUESTION:**

* Which function is used to perform some action when the object is to be destroyed?

1. Finalize () 3. Main ()
2. Delete () 4. None of them

Answer: option 1

**SUMMERY:**

* Which of the following statements are incorrect?

1. Default constructor is called at the time of object declaration.
2. Constructor can be parameterized.
3. Finalize () method is called when an object goes out of scope and is no longer needed.
4. Finalize () method must be declared protected.

Answer: option 3

* Which of these statements is incorrect?

1. By multithreading CPU idle time is minimized, and we can take maximum use of it.
2. By multitasking CPU idle is minimized, and we can take maximum use of it.
3. Two threads in java can have the same priority.
4. A thread can exist only in two states, running and blocked.

Answer: option 4