Few Things to Remember

Definitions, Tips & Rules related to Java OOPS

Class & Constructor

- *A class in Java can contain: fields, methods, constructors, blocks, nested class and interface.*
- *Object is an instance of a class.*
- *Constructor in java is a special type of method that is used to initialize the object.*
- *If there is no constructor in a class, compiler automatically creates a default constructor.*
- *There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.*
- *A constructor can perform other tasks instead of initialization like object creation, starting a thread, calling method etc.*
- *You can perform any operation in the constructor as you perform in the method.*

static Keyword

- *The static can be: variable (class variable), method (class method), block & nested class.*
- *Java static property is shared to all objects.*
- *A static method belongs to the class rather than object of a class.*
- *A static method can be invoked without the need for creating an instance of a class.**
- *A static method can access static data member and can change the value of it.*
- *The static method can not use non static data member or call non-static method directly.*
- *this and super cannot be used in static context.*
- *The main method is static because object is not required to call static method if it were non-static method, jvm create object first then call main() method that will lead the problem of extra memory allocation.*
- *A static block is used to initialize the static data member. It is executed before main method at the time of classloading.*

this Keyword

- *In Java, this is a reference variable that refers to the current object.*
- *Call to this() must be the first statement in constructor.*
- *this can be used to: refer current class instance variable, invoke current class method and constructor.*

- *this canpassed as an argument in the method and constructor call.*
- *this can be used to return the current class instance from the method.*
- *It is better approach to use meaningful names for variables. So we use same name for instance variables and parameters in real time, and always use this keyword.*

Inheritance

- *Inheritance (IS-A) is a mechanism in which one object acquires all the properties and behaviors of parent object.*
- *The extends keyword indicates that you are making a new class that derives from an existing class.*
- *Multiple inheritance is not supported in Java through class. We can use Interface to perform it.*
- *To reduce the complexity and simplify the language, multiple inheritance is not supported in java.*
- *If a class have an entity reference, it is known as Aggregation (HAS-A relationship).*
- *Inheritance should be used only if the relationship is-a is maintained throughout the lifetime of the objects involved; otherwise, aggregation is the best choice.*

Method Overloading

- *If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.*
- *There are two ways to overload the method in java : by changing number of arguments, by changing the data type.*
- *In Java, Method Overloading is not possible by changing the return type of the method only because of ambiguity.*
- *Compile Time Error is better than Run Time Error. So, java compiler renders compiler time error if you declare the same method having same parameters.*
- *We can also overload Java main() method, but JVM calls main() method which receives string array as arguments only.*
- *One type is promoted to another implicitly if no matching datatype is found. eg. byte can be promoted to short, int, etc.*
- *If there are no matching type arguments method, and each method promotes similar number of arguments, there will be ambiguity.*
- *One type is not de-promoted implicitly for example double cannot be depromoted to any type implicitly.*

Method Overriding

- *If subclass (child class) has the same method as declared in the parent class, it is known as method overriding.*

- *Method must have same name and parameters as in the parent class for overriding.*
- *Method overriding is used to provide specific implementation of a method that is already provided by its super class. Also used for runtime polymorphism.*
- *We cannot override static method (not also main method) because static method is bound with class whereas instance method is bound with object. Static belongs to class area and instance belongs to heap area.*
- *Method Overriding with Access Modifier: if you are overriding a method, overridden method (i.e. declared in subclass) must not be more restrictive.*
- *Covariant Return Type: It is possible to override method by changing the return type if subclass overrides any method whose return type is Non-Primitive but it changes its return type to subclass type.*

super Keyword

- *The super keyword is a reference variable which is used to refer immediate parent class object.*
- *super can be used to refer immediate parent class instance variable or invoke immediate parent class method and constructor.*
- *super() is added in each class constructor automatically by compiler if there is no super() or this().*

Instance initializer block

- *Instance Initializer block is used to initialize the instance data member.*
- *It is created when instance of the class is created.*
- *It runs each time when object of the class is created.*
- *It is invoked after the parent class constructor is invoked (i.e. after super() constructor call).*
- *The instance Initializer block comes in the order in which they appear.*

final Keyword

- *The final keyword in java is used to restrict the user.*
- *You cannot change the value of final variable(It will be constant).*
- *If you make any as , You cannot override a final method.*
- *If you make any class as final, you cannot extend it.*
- *Final method is inherited but you cannot override it.*
- *A final variable that is not initialized at the time of declaration is known as blank final variable.*
- *We can initialize a blank final variable only in constructor.*
- *A static final variable that is not initialized at the time of declaration is known as static blank final variable. It can be initialized only in static block.*
- *If you declare any parameter as final, you cannot change the value of it.*
- *A constructor cannot be declared final because it is never inherited.*

Runtime Polymorphism

- *Polymorphism is a concept by which we can perform a single action by different ways.*
- *There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism.*
- *We can perform polymorphism in java by method overloading and method overriding.*
- *If you overload static method in java, it is the example of compile time polymorphism.*
- *In Runtime polymorphism (Dynamic Method Dispatch), an overridden method is resolved at runtime rather than compile-time.*
- *A Virtual Method is an inheritable and overridable method for which dynamic dispatch is facilitated.*
- *All non-static, non-final and non-private methods are Virtual Methods by default.*
- *When reference variable of Parent class refers to the object of Child class, it is known as upcasting.*
- *Method is overridden not the datamembers, so runtime polymorphism can't be achieved by data members.*
- *Connecting a method call to the method body is known as binding.*
- *There are two types of binding : Static binding (early binding) and Dynamic binding (late binding).*

instanceof Keyword

- *instanceof operator is used to test whether the object is an instance of the specified type (class/subclass/interface).*
- *When Subclass type refers to the object of Parent class, it is known as downcasting.*
- *If we perform downcasting directly, there is compile error.*
- *If we perform downcasting by typecasting, ClassCastException is thrown at runtime.*
- *If we use instanceof operator, downcasting is possible!*

Abstract Class

- *A class that is declared as abstract (keyword) is abstract class. It can have abstract and non-abstract methods.*
- *Abstraction is a process of hiding the implementation details and showing only functionality to the user.*
- *There are two ways to achieve abstraction in java : Abstract class (0 to 100%) and Interface (100%).*
- *A method that is declared as abstract and does not have implementation is abstract method.*
- *Any method with a body is non-abstract method.*

- *An abstract class can have data member, abstract method, method body, constructor and even main() method.*
- *If there is any abstract method in a class, that class must be abstract.*
- *If extending any abstract class that have abstract method, we must either provide the implementation of the method or make this class abstract.*
- *Abstract class can also be used to provide some implementation of an interface. Then, the end user extending thtis abstract class is free to skip implementing that method while overriding all the methods of the interface.*

Interface

- *An interface in java is a blueprint of a class. It has static constants and abstract methods.*
- *Since Java 8, we can have method body in interface. But we need to make it default or static method.*
- *The interface is a mechanism to achieve abstraction. It represents IS-A relationship.*
- *By using interface, we can support multiple inheritance.*
- *It can be also used to achieve loose coupling (coupling is degree of direct knowledge that one element has of another).*
- *The Java compiler adds public & abstract before the interface method. Adds public, static & final before data members.*
- *A class extends another class, an interface extends another interface but a class implements an interface.*
- *Multiple inheritance is not supported by class because of ambiguity. But, supported by interface because there is no ambiguity as implementation is provided by the implementation class.*
- *An interface with no member is called marker/tagged interface. For example: Serializable, Cloneable, Remote etc.*
- *Marker interface are used to provide essential information to JVM, so that JVM may perform some useful operation.*
- *An interface can have another interface i.e. known as nested interface.*

Package

- *A java package is a group of similar types of classes, interfaces and sub-packages.*
- *Java package is used to categorize the classes and interfaces, provides access protection and removes naming collision.*
- *The package keyword is used to create a package. Package inside the package is called the subpackage.*
- *If you import a package (package.),* subpackages will not be imported.*
- *To import subpackage, use import package.classname.*
- *Use fully qualified name to access only the declared class of a package.*
- *Sequence of the program must be package then import then class.*

- *The standard of defining package is domain.company.package. eg - com.oracle.database*
- *There can be only one public class in a java source file and it must be saved by the public class name.*

Access Modifiers

- *There are two types of modifiers in java: access modifiers and non-access modifiers.*
- *There are 4 types of java access modifiers: private, default, protected & public.*
- *There are many non-access modifiers such as static, abstract, synchronized, native, volatile, transient etc.*
- *The private access modifier is accessible only within class.*
- *If you make any class constructor private, you cannot create the instance of that class from outside the class.*
- *If we don't use any modifier, it is treated as default. Default modifier is accessible only within package.*
- *A Class cannot be private or protected except nested class.*
- *The protected access modifier is accessible within package and outside the package but through inheritance only.*
- *The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.*
- *If you are overriding any method, overridden method (i.e. declared in subclass) must not be more restrictive.*

Encapsulation

- *Encapsulation is a process of wrapping code and data together into a single unit.*
- *To create a fully encapsulated class, make all data members of the class private, & use setter/getter methods to access data.*
- *By providing only setter or getter method, you can make the class read-only or write-only.*

Miscellaneous

- *The Object class is the parent class of all the classes in java by default.*
- *The Cloneable interface must be implemented by the class if we want to create a clone of an object.*
- *Wrapper class is used to convert primitive into object and object into primitive.*
- *Autoboxing and unboxing feature converts primitive into object and object into primitive automatically.*
- *There is only call by value in java, not call by reference.*
- *A method in java that calls itself is called recursive method.*