Java Descriptive Question

**** What is OOP? Feature of OOP?**

Ans: **Object Oriented Programming** is a method of implementation in which programs are organized as cooperative collection of objects, each of which represents an instance of a class, and whose classes are all members of a hierarchy of classes united via inheritance relationships. Object Oriented programs are easier to understand and less time consuming to maintain and extend.

Four principles of Object Oriented Programming are

Abstraction : Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

Encapsulation: Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

* Hides the implementation details of a class.
* Forces the user to use an interface to access data
* Makes the code more maintainable.

Inheritance: This is the process by which one object acquires the properties of another object.

Polymorphism: This is the existence of the classes or methods in different forms or single name denoting different implementations.

**** What is JVM? Describe.**

Stand for Java Virtual Machine. The JVM is the environment in which Java programs execute. It is a software that is implemented on top of real hardware and operating system.

When the source code (.java files) is compiled, it is translated into byte codes and then placed into (.class) files. The JVM executes these byte codes. So Java byte codes can be thought of as the machine language of the JVM. The JVM is an interpreter for bytecode. The JVM is a software that can be ported onto various hardware-based platforms.

**** Describe "Java is a Platform Independent Language".**

Java is compiled to an intermediate form called byte-code. A Java program never really executes natively on the host machine. Rather a special native program called the Java interpreter reads the byte code and executes the corresponding native machine instructions.

Java is platform independent as JVM compiles source code to its byte code which is then interpreted to object code. Thus any machine having a java compiler can execute that byte code. this does not depends on the hardware or the OS of the system

**** What do you mean by object and class?**

Ans: Object are all around u. Anything can be thought of as an Object. An object is an instance of a class. An Object **is** a module that has both state and behaviour.

A Class **is** nothing but a blue print for creating different objects which **defines** the shape and nature of an object. Class is a term that **describes** a specification for a collection of objects with common properties. A Class **is** a Template for an object that **defines** a new data type.

**** What do you mean by inheritance?**

Ans: Inheritance can be defined as the process where one object **acquires** the properties of another. With the use of inheritance the information is made manageable in a hierarchical order. A very important fact to remember is that Java only supports only single inheritance. This means that a class cannot extend more than one class.

**** What do you mean by 'super' and 'this' keyword?**

Ans: 'this' & 'super' is a keyword. "**this**" is used to invoke a constructor of *a same class*, its pointing the same class object. "**super**" is used to invoke a super class constructor and accessing the super class constructor.

**** Write down the difference between overloading and overridden.**

Ans: Overriding - *same method names* with *same arguments* and *same return types* associated in a class and its **subclass.**

Rules for method overriding:

- 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.
- The access level cannot be more restrictive than the overridden method's access level. For **Example**: if the super class method is declared public then the overridding method in the sub class cannot be either private or public. However the access level can be less restrictive than the overridden method's access level.
- Instance methods can be overridden only if they are inherited by the subclass.

- A method declared final cannot be overridden.
- A method declared static cannot be overridden but can be re-declared.
- If a method cannot be inherited then it cannot be overridden.
- A subclass within the same package as the instance's superclass can override any superclass method that is not declared private or final.
- A subclass in a different package can only override the non-final methods declared public or protected.
- An overriding method can throw any uncheck exceptions, regardless of whether the overridden method throws exceptions or not. However the overridden method should not throw checked exceptions that are new or broader than the ones declared by the overridden method. The overriding method can throw narrower or fewer exceptions than the overridden method.
- Constructors cannot be overridden.

Overloading - *same method name* with *different arguments, may or may not be same return type* written in the **same class itself**.

****Write down the difference between interface and abstract class?**

Ans: interface: An interface is a collection of abstract methods. An interface is not a class. Writing an interface is similar to writing a class, but they are two different concepts. A class **describes** the attributes and behaviors of an object. An interface **contains** behaviors that a class implements. A Class uses the **implements** keyword to implement an interface.

An interface is similar to a class in the following ways:

- An interface can contain any number of methods.
- An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
- The byte code of an interface appears in a **.class** file.

However, an interface is different from a class in several ways, including:

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

Abstraction refers to the ability to make a class abstract in OOP. An abstract class is one that cannot be instantiated. All other functionality of the class still exists, and its fields, methods, and constructors are all accessed in the same manner. You just cannot create an instance of the abstract class.

**** What do you mean by logical(&) and short circuit(&&) operator?**

Ans: When & is used it will evaluate both the expressions regardless of the fact that it finds first expression as FALSE and only then will it give an answer. Whereas if && was used in place of & , after it had evaluated first expression and had found result of first expression as FALSE, it would not have evaluated second expression.

**** Write down seven keywords in java?**

Ans: Int, Boolean, Static, final, Abstract, Synchronized, default, byte, long, try, catch

**** What do you mean by indexOf and charAt()?**

Ans: charAt

public char **charAt**(int index)

Returns the character at the specified index. An index ranges from 0 to length() - 1. The first character of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

Specified by: charAt in interface CharSequence

Parameters: index - the index of the character.

Returns: The character at the specified index of this string. The first character is at index 0.

Throws: IndexOutOfBoundsException - if the index argument is negative or not less than the length of this string.

indexOf

public int indexOf(int ch)

Returns the index within this string of the first occurrence of the specified character. If a character with value ch occurs in the character sequence represented by this String object, then the index of the first such occurrence is returned -- that is, the smallest value k such that: this.charAt(k) == ch

is true. If no such character occurs in this string, then -1 is returned.

Parameters: ch - a character.

Returns: the index of the first occurrence of the character in the character sequence represented by this object, or -1 if the character does not occur.

Another form of indexOf( ):

public int indexOf(int ch,int fromIndex)

Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index. If a character with value ch occurs in the character sequence represented by this String object at an index no smaller than fromIndex, then the index of the first such occurrence is returned--that is, the smallest value k such that:

(this.charAt(k) == ch) && (k >= fromIndex) is true.

If no such character occurs in this string at or after position fromIndex, then -1 is returned. There is no restriction on the value of fromIndex. If it is negative, it has the same effect as if it were zero: this entire string may be searched. If it is greater than the length of this string, it has the same effect as if it were equal to the length of this string: -1 is returned.

Parameters: ch - a character.

fromIndex - the index to start the search from.

Returns: the index of the first occurrence of the character in the character sequence represented by this object that is greater than or equal to fromIndex, or -1 if the character does not occur.

**** Write down the difference between equels() and "= ="?**

Ans: The **equals()** method and = = operator perform two different operations. The equals() method **compares** *the character inside a String object*. The = = operator **compares** *two object reference* to see whether they refer to the same instance.

**** Write down the difference between throw and throws?**

Ans: throw is used to throw an exception manually where as **throws** is used in the case of checked exceptions to reintimate the compiler that we have handled the exception. so throws is to be used at the time of defining a method and also at the time of calling that function which rises an checked exception.

**** Do you mean by static variable and instance variable and local variable?**

Static variable has only one copy for all the methods in class while instance variable has many copies. Class can access only static variable while object can access both class and instance variable.Class variables are variables declared with in a class, outside any method, with the **static** keyword.

The variable that is declared in method, constructor, or any block is called **local variable**. Access modifier can not be used for local variable. There is no default value and should be assigned initial value. It is implemented at stack level internally.

****What do you understand by StringBuffer Class? Describe Some method of string buffer class?**

Ans: StringBuffer Class:

StringBuffer class is a mutable class unlike the String class which is immutable. Both the capacity and character string of a StringBuffer Class. StringBuffer can be changed dynamically. String buffers are preferred when heavy modification of character strings is involved (appending, inserting, deleting, modifying etc).

Strings can be obtained from string buffers. Since the StringBuffer class does not override the equals() method from the Object class, **contents of string buffers should be converted to String objects for string comparison.**

A StringIndexOutOfBoundsException is thrown if an index is not valid when using wrong index in String Buffer manipulations

The following program explains the usage of the some of the basic StringBuffer methods like ;

1. **capacity**(): Returns the current capacity of the String buffer.
2. **length**(): Returns the length (character count) of this string buffer.
3. **charAt**(int index): The specified character of the sequence currently represented by the string buffer, as indicated by the index argument, is returned.
4. **setCharAt**(int index, char ch): The character at the specified index of this string buffer is set to ch
5. **toString**():Converts to a string representing the data in this string buffer
6. **insert**(int offset, char c): Inserts the string representation of the char argument into this string buffer.

Note that the StringBuffer class has got many overloaded 'insert' methods which can be used based on the application need.

7. **delete**(int start, int end): Removes the characters in a substring of this StringBuffer
8. **replace**(intstart, int end, String str): Replaces the characters in a substring of this StringBuffer with characters in the specified String.
9. **reverse**(): The character sequence contained in this string buffer is replaced by the reverse of the sequence.
10. **append**(String str): Appends the string to this string buffer.

Note that the StringBuffer class has got many overloaded 'append' methods which can be used based on the application need.

11. **setLength**(int newLength): Sets the length of this String buffer.

**** What do you mean by type casting?**

Ans: Type Casting is used to convert the value of one type to another. There are two types of typecasting.

Explicit casting: Explicit casting **is** the process in which the complier are specifically informed to about transforming the object.

Example

long i = 700.20;

Implicit casting: Implicit casting **is** the process of simply assigning one entity to another without any transformation guidance to the compiler. This type of casting is not permitted in all kinds of transformations and may not work for all scenarios.

Example

int i = 1000;

long j = i; //Implicit casting

**** Write down the difference between finalize () and finally?**

final: final keyword can be applied to vairables, method, class. final variable - You can't reassign/modify values to the variables. final class- You can not extends(inherit) the class. final method- You can not override the final methods.

finally: is the last clause in a try...catch block. It is a block of statements that is executed irrespective if or if not an exception was caught in the preceding try block. Each try contain only one finally blocks not more than one.

finalize(): This is method used to release the occupied memory /any expensive resources including native peer objects,file/device/database connections. finally method must be protected or public otherwise compile time error.

**** What do you mean by Recursion?**

Ans: Recursion is the process of defining something in terms of itself. As it relates to java, recursion is the attribute that allows a method to call itself. A method that calls itself is said to be recursive.

**** What do you mean by parameter and argument?**

Ans: A ***parameter*** **defines** the type of value that **can be passed** to the method when it is called. A parameter has a name and a type that **appears** in the parameter list in the definition of a method.

An ***argument*** is a value that is passed to a method when it is executed. The value of the arguments is referenced by the parameter name during execution of the method.

**** What do you mean by continue and break?**

Ans: A break; statement results in the termination of the statement to which it applies (switch,for, do, or while). A labeled **break** statement can be used to exit from any block, whereas an unlabeled **break** can only be used inside a loop control structure or switch structure.

· An unlabeled break terminates the innermost for, while, do or switch.

· To terminate an outer loop, use a labeled break, where the outer loop statement is proceeded by the label name.

A continue; statement is used to end the current loop iteration and return control to the loop statement. A continue statement skips to the end of a loop's body and evaluates the boolean-expression that controls the loop. In particular, in a for loop, the increment expression is executed before examining the boolean-expression.

· A continue only has meaning inside: for, while, do-while.

· If the continue keyword is followed by a label of an enclosing loop, execution skips to the end of the labeled loop.

· If no label is specified, then execution skips to the end of the loop in which the continue is located within.

**** What is difference between int and integer?**

int is a primitive datatype like char float double boolean short long. Integer is a wrapper class. Every primitive data type is having its own wrapper class like Float Double Boolean Long Character Short

int is in primitive data type and integer is an object or an wrapper class type of data and ex used in vectors.

int is a primitive data type where as Integer is a wrapper class.

int is a primitive datatype Integer isa Wrapper class .Integer is similar to object .when u have to send or get an int as an object u use Integer eg:int i 10;display(new Integer(i))

**** What do you mean by Auto-Boxing?**

Ans: *Autoboxing*, introduced in Java 5, is the automatic conversion the Java compiler makes between the primitive (basic) types and their corresponding object wrapper classes (eg, int and Integer, double and Double, etc). The underlying code that is generated is the same, but autoboxing provides a sugar coating that avoids the tedious and hard-to-read casting typically required by Java Collections, which can not be used with primitive types.

****What do you mean by 'isA' and 'hasA' relationship?**

Ans: The ***is a*** relationship is expressed with inheritance and ***has a*** relationship is expressed with composition.

IS-A Relationship:

IS-A is a way of saying : This object is a type of that object. The **extends** keyword is used to achieve inheritance. With use of extends keyword the subclasses will be able to inherit all the properties of the superclass except for the private properties of the superclass. The **implements** keyword is used by classes to inherit from interfaces. Interfaces can never be extended.

is a --- House is a Building

```
public class Animal{
}
public class Mammal extends Animal{
}
public class Reptile extends Animal{
}
public class Dog extends Mammal{
}
```

has a -- House has a bathroom

```
public class Vehicle{}
public class Speed{}
public class Van extends Vehicle{
        private Speed sp;
}
```

HAS-A relationship:

These relationships are mainly based on the usage. This determines whether a certain class **HAS-A** certain thing. This relationship helps to reduce duplication of code as well as bugs.

This shows that class Van HAS-A Speed. By having a separate class for Speed we do not have to put the entire code that belongs to speed inside the Van class., which makes it possible to reuse the Speed class in multiple applications.

What do you understand by Final Class & method?

Ans: Final Class

Java allows us to declare our class as final; that is, the class declared as final cannot be subclassed. There are two reasons why one wants to do this: security and design purpose.

- Security: To subvert systems, One of the mechanism hackers do use is, create subclasses of a class and substitute their class for the original. The subclass looks same as the original class but it will perform vastly different things, causing damage or getting into private information possibly. To prevent this subversion, you should declare your class as final and prevent any of the subclasses from being created.
- Design: Another reason us to declare a class as final is object-oriented design reason. When we feel that our class is "perfect" and should not have subclasses further. We can declare our class as final.

**** What are methods and how are they defined? Describe main method.**

Ans: **Methods** are functions that operate on instances of classes in which they are defined. Objects can communicate with each other using methods and can call methods in other classes. Method definition has four parts. They are name of the method, type of object or primitive type the method returns, a list of parameters and the body of the method. A method's signature is a combination of the first three parts mentioned above.

main( ) method is called the brain of the Java application. It is necessary to specify the name of the class which is required to run while running a Java application using the Java interpreter. Interpreter will do invokes the main( ) method defined in the class. The main( ) method will control the program flow, allocates all the resources which are needed, and will run any of the methods that do provide the functionality to the application.

**** What do you mean by Exception Handling? What is the purpose of Exception Handling?**

Ans: An exception is an event which occurs during execution of the program which disrupts the normal flow of the instructions. Different types of errors can cause exceptions: problems which range from serious hardware errors, such as hard disk crash, to the simple programming errors, like trying to access the out-of-bounds array element. When such error occurs within a Java method, the method will create an exception object and hands it to the runtime system. The exception object do contains the information about exception including its type and state of the program when error occurred. Runtime system is then responsible to find some code to handle error. In the Java terminology, creating exception object and handing it to runtime system is called "throwing an exception".

An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons, including the following:

- A user has entered invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications, or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

To understand how exception handling works in Java, you need to understand the three categories of exceptions:

- **Checked exceptions:** Achecked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For **Example**, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.
- **Runtime exceptions:** A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compliation.
- **Errors:** These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For **Example**, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

**** What is the advantage of package? Write down the default package.**

Ans: A Package **is** a collection of classes and interfaces that **provides** a high-level layer of access protection and name space management. In short it is a folder to organize your classes and interfaces

Packages are useful to :

- ✓ organize classes into smaller units so that it is easy to locate
- ✓ helps to avoid naming conflicts
- ✓ protect your classes, data and methods in a larger way on a class to class basis identify classes with the help of the package names Can also contain other packages representing a smaller, more specific grouping of classes to make a class member of the package - begin your code with a package declaration

Default package is java.lang

What is the difference between Array and vector?

Ans: Array is a set of related data type and static *whereas* **vector** is a growable array of objects and dynamic.

What is the difference between exception and error?-

Ans: The exception class defines mild error conditions that your program encounters. Exceptions can occur when trying to open the file, which does not exist, the network connection is disrupted, operands being manipulated are out of prescribed ranges, the class file you are interested in loading is missing. The error class defines serious error conditions that you should not attempt to recover from. In most cases it is advisable to let the program terminate when such an error is encountered.

What is Garbage Collection and how to call it explicitly?-

Ans: When an object is no longer referred to by any variable, java automatically reclaims memory used by that object. This is known as garbage collection. System. gc() method may be used to call it explicitly.

What are Transient and Volatile Modifiers?-

Ans: Transient: The transient modifier applies to variables only and it is not stored as part of its object's Persistent state. Transient variables are not serialized.

Volatile: Volatile modifier applies to variables only and it tells the compiler that the variable modified by volatile can be changed unexpectedly by other parts of the program

What is difference between overloading and overriding?

Ans: a) In overloading, there is a relationship between methods available in the same class *whereas* in overriding, there is relationship between a superclass method and subclass method.

b) Overloading does not block inheritance from the superclass *whereas* overriding blocks inheritance from the superclass.

c) In overloading, separate methods share the same name *whereas* in overriding, subclass method replaces the superclass.

d) Overloading must have different method signatures *whereas* overriding must have same signature.

What is the difference between this() and super()?

Ans: **this()** can be used to invoke a constructor of the same class *whereas* **super()** can be used to invoke a super class constructor.

What is the difference between superclass and subclass?-

Ans: A super class is a class that is inherited *whereas* sub class is a class that does the inheriting.

What modifiers may be used with top-level class?-

Ans: public, **abstract** and **final** can be used for top-level class.

What is interface and its use?

Ans: Interface is similar to a class which may contain method's signature only but not bodies and it is a formal set of method and constant declarations that must be defined by the class that implements it.

Interfaces are useful for:

a) Declaring methods that one or more classes are expected to implement

b) Capturing similarities between unrelated classes without forcing a class relationship.

c) Determining an object's programming interface without revealing the actual body of the class.

What is an abstract class?

Ans: An abstract class is a class designed with implementation gaps for subclasses to fill in and is deliberately incomplete.

What is the difference between Integer and int?-

Ans: a) **Integer is** a class defined in the java.lang package, *whereas* **int** is a primitive data type defined in the Java language itself. Java does not automatically convert from one to the other.

b) **Integer** can be used as an argument for a method that requires an object, *whereas* **int** can be used for calculations.

What is the difference between String and String Buffer?-

Ans: a) String objects are **constants** and **immutable** *whereas* StringBuffer objects are not.

b) String class **supports** constant strings *whereas* StringBuffer class supports growable and modifiable strings.

What are wrapper classes?

Ans: Java provides specialized classes corresponding to each of the primitive data types. These are called wrapper classes. They are e.g. Integer, Character, Double etc.

Why do we need wrapper classes?

Ans: It is sometimes easier to deal with primitives as objects. Moreover most of the collection classes store objects and not primitive data types. And also the wrapper classes provide many utility methods also. Because of these reasons we need wrapper classes. And since we create instances of these classes we can store them in any of the collection classes and pass them around as a collection. Also we can pass them around as method parameters where a method expects an object.

What are runtime exceptions?

Ans: Runtime exceptions are those exceptions that are thrown at runtime because of either wrong input data or because of wrong business logic etc. These are not checked by the compiler at compile time.

**** How many ways we can declare an array and what are they?**

Ans: We can declare arrays in the following ways…

```
float[] f1 = new float[3];
float f3[] = new float[3];
float f5[] = { 1.0f, 2.0f, 2.0f };
float f4[] = new float[] { 1.0f. 2.0f. 3.0f};
```

**** What Do you mean by enum?**

Ans: An *enum type* is **a type** whose *fields* consist of a fixed set of constants. Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST) and the days of the week. Because they are constants, the names of an enum type's fields are in uppercase letters. In the Java programming language, you define an enum type by using the enum keyword. For example, you would specify a days-of-the-week enum type as:
public enum Day {SUNDAY, MONDAY, TUESDAY, WEDNESDAY,THURSDAY, FRIDAY, SATURDAY }

**** Write down the difference between while and do while?**

Ans: Loops execute a block of code a specified number of times, or while a specified condition is true
do...while:The do...while loop is a variant of the while loop. This loop will execute the block of code at least ONCE, and then it will repeat the loop as long as the specified condition is true.

The while loop loops through a block of code while a specified condition is true.

**** What do you mean by Assertion? Declare an assertion statement?**

Ans: Assert is used during program development to create an assertion which is a condition that should be true during the execution of program. Assertions are often used during testing to verify that some expected condition is actually met.

**** What do you mean by narrowing and widening? Give an example of them.**

3.9.2 Widening and Narrowing

Conversion from a number with a smaller range of magnitude (like `int` to `long` or `long` to `float`) is called *widening*. Conversely, conversion where there is the possibility of losing information about the magnitude of the number (like `long` to `int` or `double` to `long`) is called *narrowing*.

Narrowing operations may lose information about how big the number is. For example, converting the `float` 3.4028235e38 to `long` yields `-9223372036854775808`, a very different number. However, narrowing is not necessarily accompanied by the loss of information. For example, there is an exact representation of the double 1.0 in `int`: the `int` 1. Nor is widening necessarily without information loss. For example, converting the int `2147483647` to `float` produces `2.14748365e9`, or 2,147,483,650. The difference is usually small, but it may be significant.

The goal of widening conversions is to produce no change in the magnitude of the number while preserving as much of the precision as possible. With narrowing conversions, some information may be lost, but the nearest representation is found whenever possible.

**** Do you think java support multiple-inheritance? If no what we use for its substitution an why?**

Ans: NO. We use interface.

**** Which class is the mother of all class? Write down the difference between wrapper class and data type?**

Ans: Object class is called the mother class of all class.

**** What do you mean by tokenizing?**

Ans: The split( ) method in the String class is specifically for splitting into tokens. This is done in a single step, returning all the tokens from a string as an array of String objects. This procedure is known as Tokenizing.

**** Describe Standard Package?**

Ans: All of the standard classes that are provided with java are stored in a standard package. There is a substantial and growing list of standard packages.Some of those are as follows:

java.lang	java.io	java.nio	java.nio. channels	java.awt	javax.swing java.awt.evnt	java.awt.geom	java.applet	java.util