1. What is an Object?

Any real world entity is an object. eq. car, laptop, student, employee, pen.

java object

An object has state, which is also called as properties. A student object has properties as name, studentId, city, phone,department.A car object has properties like model, brand, price.

An object can have behavior, which is called as methods. A student object can have methods as printDetails(), getGrade(). A car object can have methods like getMileage(), printAccessories()

2. What is a Class?

A class is a blueprint of an object. A class is a collection of objects with common properties , behavior , relationship and semantics. A collection of student object becomes a Student class and collection of car objects becomes Car class. An object is an instance of a class.

java class

3. What are instance variables?

The properties of an object are called instance variables. eg. A Student class can have instance variables as name, studentId. In case of the instance variable, the variable name is same but the values are different for every object.

4. What is a static variable?

A static variable is one, which is common to all the objects.Only one copy of the variable is available and shared by all the objects eg. A Student class can the teacher name as a static variable.

In case of static variable, the value is same for all the objects. It can be called using classname.variablename

5. What is a Constructor?

A constructor is used to initialize the instance variables. It has the same name as that of the class and does not have a return type.

The syntax for the constructor is access-specifier class-name

Example

public Student(){}

1

2

3

public Student(){}

What is a default constructor?

A default constructor is one without parameters

Can a class have more than one constructor?

Yes. A class can have more than one constructor taking different parameters. This concept is called constructor overloading

What will happen if I have not added any constructor to my class?

If you have not added any constructor, the compiler will add the default constructor automatically

Can a constructor be private?

Yes. But you can create the object only in the same class.If needed the object can be returned using a static method.

Show an example of constructor with and without parameters

Example

class Student{

String name;

int studentId; //instance variables

//default constructor

public Student(){

this.name = "Ram";

this.studentId= 123;

}

//parameterised constructor

public Student(String name, int studentId){

this.name = name;

this.studentId= studentId;

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

class Student{

String name;

int studentId; //instance variables

//default constructor

public Student(){

this.name = "Ram";

this.studentId= 123;

}

//parameterised constructor

public Student(String name, int studentId){

this.name = name;

this.studentId= studentId;

}

}

What will be in the first line of any constructor?

The first line of any constructor will have either this() or super() keyword.

How will you create an object?

Use the new keyword to create an object. The new keyword() is used to allocate memory.

Example

Student student = new Student();

where new Student() is the object,

student is just a reference variable pointing to a student object.

What is a method?

The methods are operations done on that object. It can be with or without return type and with or without parameter. The syntax for a method is

access-specifier return-type method-name(parameter-list)Example

int calcSum(int a, int b){

return a+b;

}

void printDetails(String name){

System.out.println("Welcome "+name);

}

1

2

3

4

5

6

7

8

int calcSum(int a, int b){

return a+b;

}

void printDetails(String name){

System.out.println("Welcome "+name);

}

What will happen if the instance variable name and the local variable name are same?>

In this case the local variable hides the instance variable. the instance variable will take the default values of the data type. Use ‘this’ keyword to prevent it

public class Student{

String name;

int studentId;

public Student(String name, int studentId) {

this.name= name;

this.studentId= studentId;

}

}

1

2

3

4

5

6

7

8

9

10

public class Student{

String name;

int studentId;

public Student(String name, int studentId) {

this.name= name;

this.studentId= studentId;

}

}

What is the meaning of this keyword?

‘this’ refers to the current object.

To learn about the top questions and answers in overloading click here

What restrictions are placed on method overloading?

Two methods may not have the same name and argument list but different return types.

What is the difference between String and StringBuffer?

String objects are immutable whereas StringBuffer objects are not. StringBuffer unlike Strings support growable and modifiable strings.

Can a private method of a superclass be declared within a subclass?

Sure. A private field or method or inner class belongs to its declared class and hides from its subclasses.

There is no way for private stuff to have a runtime overloading or overriding (polymorphism) features.

What is the default value of an object reference declared as an instance variable?

null unless we define it explicitly.

What is the difference between a constructor and a method?

Or

How can a subclass call a method or a constructor defined in a superclass?

A constructor is a member function of a class that is used to create objects of that class, invoked using the new operator. It has the same name as the class and has no return type. They are only called once, whereas member functions can be called many times. A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator. Constructor will be automatically invoked when an object is created whereas method has to be called explicitly.

super.method(); is used to call a super class method from a sub class. To call a constructor of the super class, we use the super(); statement as the first line of the subclass’s constructor.

Can a top-level class be private or protected?

No. A top-level class cannot be private or protected. It can have either “public” or no modifier. If it does not have a modifier it is supposed to have a default access. If a top level class is declared as private/protected the compiler will complain that the “modifier private is not allowed here”.

Why Java does not support multiple inheritance?

Java does support multiple inheritance via interface implementation.

Where and how can you use a private constructor?

Private constructor can be used if you do not want any other class to instantiate the class. This concept is generally used in Singleton Design Pattern. The instantiation of such classes is done from a static public method.

How are this() and super() used with constructors?

this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

What is Method Overriding? What restrictions are placed on method overriding?

When a class defines a method using the same name, return type, and argument list as that of a method in its superclass, the method in the subclass is said to override the method present in the Superclass. When the method is invoked for an object of the

class, it is the new definition of the method that is called, and not the method definition from superclass.

Restrictions placed on method overriding

• Overridden methods must have the same name, argument list, and return type.

• The overriding method may not limit the access of the method it overrides. Methods may be overridden to be more public, not more private.

• The overriding method may not throw any exceptions that may not be thrown by the overridden method.

What are the Object and Class classes used for? Which class should you use to obtain design information about an object?

Differentiate between a Class and an Object?

The Object class is the highest-level class in the Java class hierarchy. The Class class is used to represent the classes and interfaces that are loaded by a Java program. The Class class is used to obtain information about an object’s design. A Class is only a definition or prototype of real life object. Whereas an object is an instance or living representation of real life object. Every object belongs to a class and every class contains one or more related objects.

What is a singleton class?

Or

What is singleton pattern?

This design pattern is used by an application to ensure that at any time there is only one instance of a class created. You can achieve this by having the private constructor in the class and having a getter method which returns an object of the class and creates one for the first time if its null.

What is method overloading and method overriding?

Or

What is difference between overloading and overriding?

Method overloading: When 2 or more methods in a class have the same method names with different arguments, it is said to be method overloading. Overloading does not block inheritance from the superclass. Overloaded methods must have different method signatures

Method overriding : When a method in a class has the same method name with same arguments as that of the superclass,

it is said to be method overriding. Overriding blocks inheritance from the superclass. Overridden methods must have same signature.

Basically overloading and overriding are different aspects of polymorphism.

static/early binding polymorphism: overloading

dynamic/late binding polymorphism: overriding

If a class is declared without any access modifiers, where may the class be accessed?

A class that is declared without any access modifiers is said to have package or default access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.

Does a class inherit the constructors of its superclass?

A class does not inherit constructors from any of its super classes.

Which java.util classes and interfaces support event handling?

The EventObject class and the EventListener interface support event processing

Can an object’s finalize() method be invoked while it is reachable?

An object’s finalize() method cannot be invoked by the garbage collector while the object is still reachable. However, an object’s finalize() method may be invoked by other objects.

What is the purpose of the Runtime class?

The purpose of the Runtime class is to provide access to the Java runtime system.

It returns the runtime information like memory availability.

\* Runtime.freeMemory() –> Returns JVM Free Memory

\* Runtime.maxMemory() –> Returns the maximum amount of memory that the JVM will attempt to use. It also helps to run the garbage collector

\* Runtime.gc()

What is the purpose of the System class?

The purpose of the System class is to provide access to system resources.

Can an unreachable object become reachable again?

An unreachable object may become reachable again. This can happen when the object’s finalize() method is invoked and the object performs an operation which causes it to become accessible to reachable object.

What is a bean? Where can it be used?

A Bean is a reusable and self-contained software component. Beans created using java take advantage of all the security and platform independent features of java. Bean can be plugged into any software application. Bean is a simple class which has set and get methods. It could be used within a JSP using JSP tags to use them.

What is the functionality of instanceOf() ?

instanceOf opertaor is used to check whether an object can be cast to a specific type without throwing ClassCastException.

What would happen if you say this = null?

It will come up with Error Message

“The left-hand side of an assignment must be a variable”.

I want to create two instances of a class ,But when trying for creating third instance it should not allow me to create . What i have to do for making this?

One way of doing this would be:

public class test1

static int cntr=0;

test1()

{ cntr++;

if(cntr>2)

throw new NullPointerException();//u can define a new exception // for this

public static void main(String args[])

test1 t1= new test1();

System.out.println(“hello 1″);

test1 t2= new test1();

System.out.println(“hello 2″);

test1 t3= new test1();

}

}

What is the difference between an object and an instance?

An Object May not have a class definition. eg int a[] where a is an array.

An Instance should have a class definition.

eg MyClass my=new MyClass();

my is an instance.

What is heap in Java?

It is a memory area which stores all the objects created by an executing program.

Why default constructor of base class will be called first in java?

A subclass inherits all the methods and fields (eligible one) from the base class, so base class is constructed in the process of creation of subclass object (subclass is also an object of superclass). Hence before initializing the default value of sub class the super class should be initialized using the default constructor.

What are the other ways to create an object other than creating as new object?

We can create object in different ways;

1.new operator

2.class.forName: Classname obj = Class.forName(“Fully Qualified class Name”).newInstance();

3.newInstance

4.object.clone

What is the difference between instance, object, reference and a class?

Class: A class is a user defined data type with set of data members & member functions

Object: An Object is an instance of a class

Reference: A reference is just like a pointer pointing to an object

Instance: This represents the values of data members of a class at a particular time

ava - Interview Questions and Answers on Classes and Objects

Q1. What are different types of inner classes ?

Ans. Simple Inner Class, Local Inner Class, Anonymous Inner Class , Static Nested Inner Class.

Q3. Does Constructor creates the object ?

Ans. New operator in Java creates objects. Constructor is the later step in object creation. Constructor's job is to initialize the members after the object has reserved memory for itself.

Q4. Difference between == and .equals() ?

Ans. "equals" is the member of object class which returns true if the content of objects are same whereas "==" evaluate to see if the object handlers on the left and right are pointing to the same object in memory.

Q5. There are two objects a and b with same hashcode. I am inserting these two objects inside a hashmap.

hMap.put(a,a);

hMap.put(b,b);

where a.hashCode()==b.hashCode()

Now tell me how many objects will be there inside the hashmap?

Ans. There can be two different elements with the same hashcode. When two elements have the same hashcode then Java uses the equals to further differentation. So there can be one or two objects depending on the content of the objects.

Q6. What are the common uses of "this" keyword in java ?

Ans. "this" keyword is a reference to the current object and can be used for following -

1. Passing itself to another method.

2. Referring to the instance variable when local variable has the same name.

3. Calling another constructor in constructor chaining.

Q7. What are wrapper classes ?

Ans. They are wrappers to primitive data types. They allow us to access primitives as objects.

Q8. What one should take care of, while serializing the object?

Ans. One should make sure that all the included objects are also serializable. If any of the objects is not serializable then it throws a NotSerializable Exception.

Q9. Describe what happens when an object is created in Java ?

Ans. 1. Memory is allocated from heap to hold all instance variables and implementation-specific data of the object and its superclasses. Implemenation-specific data includes pointers to class and method data.

2. The instance variables of the objects are initialized to their default values.

3. The constructor for the most derived class is invoked. The first thing a constructor does is call the constructor for its superclasses. This process continues until the constructor for java.lang.Object is called,

as java.lang.Object is the base class for all objects in java.

4. Before the body of the constructor is executed, all instance variable initializers and initialization blocks are executed. Then the body of the constructor is executed. Thus, the constructor for the base class completes first and constructor for the most derived class completes last.

Q10. What are the methods of Object Class ?

Ans. clone() - Creates and returns a copy of this object.

equals() - Indicates whether some other object is "equal to" this one.

finalize() - Called by the garbage collector on an object when garbage collection determines that there are no more references to the object

getClass() - Returns the runtime class of an object.

hashCode() - Returns a hash code value for the object.

toString() - Returns a string representation of the object.

notify(), notifyAll(), and wait() - Play a part in synchronizing the activities of independently running threads in a program.

Q11. Which access specifier can be used with Class ?

Ans. For top level class we can only use "public" and "default". We can use private with inner class.

Q12. Difference between Abstract and Concrete Class ?

Ans. Abstract classes are only meant to be sub classed and not meant to be instantiated whereas concrete classes are meant to be instantiated.

Q13. What is a Final Class ?

Ans. A Class that cannot be sub classed.

Q14. What is an immutable class ?

Ans. Class using which only immutable (objects that cannot be changed after initialization) objects can be created.

Q15. Difference between object instantiation and construction ?

Ans. Though It's often confused with each other, Object Creation ( Instantiation ) and Initialization ( Construction ) are different things in Java. Construction follows object creation.

Object Creation is the process to create the object in memory and returning its handler. Java provides New keyword for object creation.

Initialization is the process of setting the initial / default values to the members. Constructor is used for this purpose. If we don't provide any constructor, Java provides one default implementation to set the default values according to the member data types.

Q16. Difference between nested and inner classes ?

Ans. Inner classes are non static nested classes.

Q17. When do you get ClassCastException?

Ans. As we only downcast class in the hierarchy, The ClassCastException is thrown to indicate that code has attempted to cast an object to a subclass of which it is not an instance.

Q18. Difference Between this() and super() ?

Ans. 1.this is a reference to the current object in which this keyword is used whereas super is a reference used to access members specific to the parent Class.

2.this is primarily used for accessing member variables if local variables have same name, for constructor chaining and for passing itself to some method whereas super is primarily used to initialize base class members within derived class constructor.

Q19. What is the benefit of inner / nested classes ?

Ans. You can put related classes together as a single logical group.

Nested classes can access all class members of the enclosing class, which might be useful in certain cases.

Nested classes are sometimes useful for specific purposes. For example, anonymous inner classes are useful for writing simpler event-handling code with AWT/Swing.

Q20. Explain Static nested Classes ?

Ans. The accessibility (public, protected, etc.) of the static nested class is defined by the outer class.

A static nested class is not an inner class, it's a top-level nested class.

The name of the static nested class is expressed with OuterClassName.NestedClassName syntax.

When you define an inner nested class (or interface) inside an interface, the nested class is declared implicitly public and static.

Static nested classes can be declared abstract or final.

Static nested classes can extend another class or it can be used as a base class.

Static nested classes can have static members.

Static nested classes can access the members of the outer class (only static members, obviously).

The outer class can also access the members (even private members) of the nested class through an object of nested class. If you don’t declare an instance of the nested class, the outer class cannot access nested class elements directly.

Q21. Explain Inner Classes ?

Ans. The accessibility (public, protected, etc.) of the inner class is defined by the outer class.

Just like top-level classes, an inner class can extend a class or can implement interfaces. Similarly, an inner class can be extended by other classes, and an inner interface can be implemented or extended by other classes or interfaces.

An inner class can be declared final or abstract.

Inner classes can have inner classes, but you’ll have a hard time reading or understanding such complex nesting of classes.

Q22. Explain Method Local Inner Classes ?

Ans. You can create a non-static local class inside a body of code. Interfaces cannot have local classes, and you cannot create local interfaces.

Local classes are accessible only from the body of the code in which the class is defined. The local classes are completely inaccessible outside the body of the code in which the class is defined.

You can extend a class or implement interfaces while defining a local class.

A local class can access all the variables available in the body of the code in which it is defined. You can pass only final variables to a local inner class.

Q23. Explain about anonymous inner classes ?

Ans. Anonymous classes are defined in the new expression itself, so you cannot create multiple objects of an anonymous class.

You cannot explicitly extend a class or explicitly implement interfaces when defining an anonymous class.

An anonymous inner class is always created as part of a statement; don't forget to close the statement after the class definition with a curly brace. This is a rare case in Java, a curly brace followed by a semicolon.

Anonymous inner classes have no name, and their type must be either a subclass of the named type or an implementer of the named interface

Q24. Difference between Class#getInstance() and new operator ?

Ans. Class.getInstance doesn't call the constructor whereas if we create an object using new operator , we need to have a matching constructor or copiler should provide a default constructor.

Q25. Can we create an object if a Class doesn't have any constructor ( not even the default provided by constructor ) ?

Ans. Yes , using Class.getInstance.

Q26. What are different ways of object creation in Java ?

Ans. Using new operator - new xyzClass()

Using factory methods - xyzFactory.getInstance( )

Using newInstance( ) method - (Class.forName(“xyzClass”))emp.newInstance( )

By cloning an already available object - (xyzClass)obj1.clone( )