**Oops Concepts**

1. **Data Hiding:**

Outside person cannot access our internal data directly or our internal data should not go out directly this OOP ffeature is called as data hiding.

After validation or authentication only outside person can access our internal data.

Example:

1. After providing proper user name and password we can able to access our gmail inbox information. Even though we are valid customer of the bank we can able to access our account information and we cannot access others account information.

We can achieve or implement data hiding by declaring data member (variable) as private

Public class Account

{

Private double balance;

Public double getBalance ()

{

//authentication or validation

return balance;

}

}

The biggest advantage of data hiding is **security.**

**Note:** It is highly recommended to declare data member (variable) as **private**

1. **Abstraction: IMP**

Hiding internal implementation and just highlight set of services we are offering is a concept of abstraction.

**How it is implemented in java?**

By using **GUI** andby using interface and Abstract classes we can be implement abstraction.

**Example:**

Through bank ATM GUI screen, bank people are highlighting the set of services they are offering like withdraw, get balance, mini statement etc without highlighting internal implementation. So when I press get balance I am getting functionality of knowing my balance but i no need to know what happened internally to return my balance. Like which data base is queried which algorithm is used to encrypt my pin number etc.

**Advantages**:

1. **Security**: Outside person does not know our implementation so we are getting security.
2. **Improves Maintainability of the application**
3. **Enhancement becomes easy:** say we have implemented get balance feature in java and due to some reasons we would like replace it with other language say C# in that case it does not affect the end user who is using for him get balance is still a feature to know his account balance
4. **Improves easiness to use system**
5. **Encapsualtion IMP**

Process of wrapping data and corresponding behaviour into a single unit is called as encapsulation.

**How it is implemented n java?**

Encapsulation is implemented using Class in java

Example:

Public Class Account

{

Private double balance;

Public double getBalance()

{

//validation

Return balance;

}

Public void setBalance(double balance)

{

This.balance = balance;

}

}

Advantages:

1. We can achieve Security
2. It improves maintainability of the application.
3. Easy to enhance.

The main advantage of encapsulation is we can achieve security but the main disadvantage of encapsulation is it increases length of the code and slows down execution.

**Tightly Encapsulated class:**

If each and every variable of the class is private then such a class is called as tightly encapsulated class.

Which of the following classes are tightly encapsulated?

Class A // Tightly encapsulated : Yes

{

Private int x = 10;

}

Class B extends A // Tightly encapsulated : No

{

int y = 10;

}

Class C extends A // Tightly encapsulated : Yes

{

Private int z=10;

}

Which of the following classes are tightly encapsulated.

Class A // Tightly encapsulated : No

{

Int x = 10;

}

Class B extends A // Tightly encapsulated : No

{

Private int y = 10;

}

Class C extends B // Tightly encapsulated : No

{

Private int z=30;

}

If the parent class is non-tightly encapsulated then its child automatically becomes non tightly encapsulated class.

If any component follows data hiding and abstraction such type of component is said to be encapsulated component.

Encapsulation = data hiding + abstraction

Suppose if we assume we have GUI when we click Balance enquiry internally get balance will be called and when setBalance button is clicked internally setBalance is called. So abstraction + databinding is together encapsulation

1. **Inheritance: Is -a relationship IMP**

One object acquires all the properties and behaviours of parent object.

It is also known as is –a relationship.

**How it is implemented in java?**

By using **extends** keyword we can implement is-a relation ship

Advantage:

Code Reusability.

Total java API is implemented based on inheritance concept.

The most common methods which are applicable for any java object are defined in Object class and hence every class in java is the child class of object either directly or indirectly so that object class methods by default available to every java class without rewriting due to this object class access root for all java classes.

Throwable class defines the most common methods which are required for every Exception and Error classes. Hence this class acts as root for java exception hierarchy.

Multiple inheritance:

Java class cannot extend more than one class at a time. Java won’t support multiple inheritance in classes

Class A extends B,C // compile time error

Note : If our class does not extend any other class then only our class is direct child class of object.

Class A

{

} //Child of Object

Class A extends B

{

} // here our class is indirect child class of object A is child of B B is child of Object Here it is multi level inheritance.

So Java does not support multiple inheritances with respect to classes at any cost

**Why Java won’t support for multiple inheritances?**

.

1. **Polymorphism IMP**

An object behaves differently in different situations. There are two types of polymorphism –

**Compile time polymorphism and runtime polymorphism.**

Compile time polymorphism is achieved by method **overloading**.

Method overloading is the case where multiple methods are having same name but different number or type of arguments. Here compiler will be able to identify the method to invoke at compile time; hence it’s called compile time polymorphism.

Runtime polymorphism is implemented method **overriding** because subclass has to override the superclass method for runtime polymorphism. If we are working in terms of superclass, the actual implementation class is decided at runtime. Compiler is not able to decide which class method will be invoked. This decision is done at runtime, hence the name as runtime polymorphism or dynamic method dispatch.

<http://www.journaldev.com/12496/oops-concepts-java-object-oriented-programming-concepts>

1. **Has-A relationship IMP**

It is also known as composition or aggregation.

There is no specific keyword to implement has-a relation but most of the times. We are depending on new Keyword

Main advantage of has a relationship is reusability of the code.

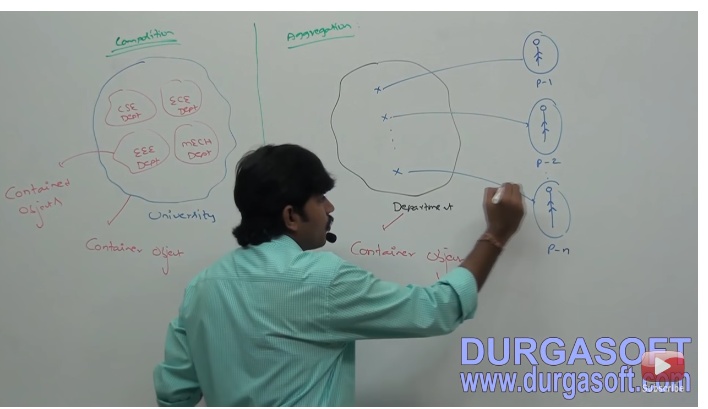
Example:

Class Car

{

Engine e = new Engine (); // Car has a engine reference.

}



Difference between Composition and Aggregation.

**Composition**: Without existing container object if there is no chance of existing contained objects then container and contained objects are strongly associated and this strong association is nothing but composition.

Example:

University consists of several departments without existing university there is no chance of existing department. Hence university and department are strongly associated and this strong association is called as composition.

Remember that diagram of surge soft

**Aggregation**: Without existing container object if there is a chance of existing contained object then container and contained objects are weakly associated and this weak association is called as aggregation.

Example:

Department consists of several professors without department there may be a chance of existing professor objects hence department and professor objects are weakly associated and this weak association is nothing but aggregation.

Note: In composition objections are strongly associated where as in aggregation objects are weakly associated. In composition Container object holds directly contained objects where as in aggregation container object holds just references of contained objects.

Is-a VS has-a

1. If we want all members of class then automatically we should go for Is-a relationship

Example Person class is there and Student require complete functionality of person class required for student class then it is a relationship.

1. I we want part of the functionality then we should go for has-a relationship.

Example: Marks card class many methods but sometimes Student class require only calculating average of marks then Student can contain Markscard object that is Has-a relation ship

* **Constructors**

1. **What is constructor?**

Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation.

1. **What is the purpose of default constructor?**

The default constructor provides the default values to the objects. The java compiler creates a default constructor only if there is no constructor in the class.

1. **Does constructor return any value?**

Yes, that is current instance (You cannot use return type yet it returns a value).

1. **Is constructor inherited?**

No, constructor is not inherited.

1. **Can you make a constructor final?**

No, constructor can't be final.

* **Static**

1. **What is static variable?**

Static variable gets memory only once in class area at the time of class loading.

Static variable is used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.

1. **What is static method? imp**

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.

Static method can access static data member and can change the value of it.

1. **Why main method is static?**

Because object is not required to call static method if It were non-static method, jvm creates object first then call main () method that will lead to the problem of extra memory allocation.

1. **What is static block?** Static Initialization block Need to see some practice programs also from lara

Is used to initialize the static data member

It is executed before main method at the time of class loading.

1. **Can we execute a program without main () method?**

Yes, one of the ways is static block.

1. **What if the static modifier is removed from the signature of the main method?**

Program compiles. But at runtime throws an error "NoSuchMethodError".

1. **What is difference between static (class) method and instance method? IMP**
2. A method i.e. declared as static is known as static method.

A method i.e. not declared as static is known as instance method.

1. Object is not required to call static method.

Object is required to call instance methods.

1. Non-static (instance) members cannot be accessed in static context (static method, static block and static nested class) directly.

Static and non-static variables both can be accessed in instance methods.

1. For example: public static int cube(int n){ return n\*n\*n;}

For example: public void msg(){...}.

1. **What is the difference between static method and static block? IMP**

A static method is specific to a class, not to instances of the class.

For example, if you define a static integer in your class and set it to 3, and you have a static method which returns that variable and one which sets the variable, then every instance of the class accesses the same data and it will always be equal to 3. If one instance changes that value using the static setter, then it will be changed for all instances which access it.

A static block, or a static initializer, is a section of code which is run when the class is loaded. You can initialize static variables in a static block, so that when class is first loaded, all of the static (or shared) variables are properly initialized.

1. **Can you access a non static member in static method if you need to access what to do? IMP**

**Need to create object in order to acess.**

**public** **class** A

{

**int** i;

**public** **static** **void** main(String[] args)

{

A a1 = **new** A();

**int** s = a1.*return1*(0);

}

**public** **static** **int** return1(**int** i)

{

A a2 = **new** A();

**int** s = a2.i;

**return** s;

}

}

1. **How Encapsulation is different from defining class and methods final? IMP**
2. **Static member called means it is instantiated how many times? IMP**
3. **LARA TECHNOLOGIS PROGRAM PRACTICE SPECIALL INHERITANCE AND STATIC CONCEPTS**
4. **Instance initialization block and its working? Can refer Lara notes**

* **Abstraction**

1. **What is abstraction?**

Hiding internal implementation and just highlight set of services we are offering is a concept of abstraction.

Abstraction lets you focus on what the object does instead of how it does it.

1. **What is the difference between abstraction and encapsulation?**
2. **What is abstract class?**

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

1. **Can there be any abstract method without abstract class?**

No, if there is any abstract method in a class, that class must be abstract.

1. **Can you use abstract and final both with a method?**

No, because abstract method needs to be overridden whereas you can't override final method.

1. **Is it possible to instantiate the abstract class? IMP**

No, abstract class can never be instantiated.

1. **Can Abstract class have constructor? IMP**

Yes abstract class can have constructor. But still we cannot instantiate the abstract class. The constructor of abstract class can be used to initialize the instance variables of abstract class.

1. **What is interface?**

Interface is a blueprint of a class that have static constants and abstract methods.It can be used to achieve fully abstraction and multiple inheritances.

1. **Can you declare an interface method static?**

No, because methods of an interface are abstract by default, and static and abstract keywords can't be used together.

1. **Can an Interface be final?**

No, because its implementation is provided by another class.

1. **What is marker interface?**

An interface that has no data member and method is known as a marker interface.For example Serializable, Cloneable etc.

1. **What is difference between abstract class and interface? IMP**

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| An abstract class can have instance variables. | An interface cannot have instance variables. |
| An abstract class can have method body (non-abstract methods). | Interfaces have only abstract methods. |
| An abstract class can have constructor. | Interface cannot have constructor. |
| An abstract class can have static methods. | Interface cannot have static methods. |
| You can extends one abstract class. | You can implement multiple interfaces. |

In case of Java8

Note: The main difference between abstract class and interface in Java 8 is the fact that abstract class is a class and interface is an interface.

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| Abstract class can have instance variable(state) and state can be modified by non instance methods | An interface cannot have instance variables. |
| Abstract class can have constructor | Interface cannot have constructor |

1. **Can we define private and protected modifiers for variables in interfaces?**

No, they are implicitly public.

1. **When to use an interface? IMP**

Say we have an application and that application has to provide APIs (blue print) These APIs has to be implemented by third party by fully adhering to the specification/method signature

1. **When to use abstract class over interface? IMP**

When there is a situation where we need to use non static or non final variables in class and there should be implementation for some behaviour and some behaviour needs to be implemented by somebody else we can use abstract class

Such abstract classes are similar to interfaces, except that they provide a partial implementation, leaving it to subclasses to complete the implementation. If an abstract class contains only abstract method declarations, it should be declared as an interface instead.

Multiple interfaces can be implemented by classes anywhere in the class hierarchy, whether or not they are related to one another in any way. Think of Comparable or Cloneable, for example.

By comparison, abstract classes are most commonly subclassed to share pieces of implementation. A single abstract class is subclassed by similar classes that have a lot in common (the implemented parts of the abstract class), but also have some differences (the abstract methods

1. **Can we have static main in abstract class? IMP**

Yes. We know that abstract class cannot be instantiated but we can have static methods because calling them does not need instance

1. **Can an abstract class have method code for all methods and no abstract method? IMP**
2. **If abstract and concrete subclass both have default constructors can you create an object of abstract class? IMP**

No

1. **Can interface have instance variable? IMP**
2. **When can an object reference be cast to an interface reference?**
3. **Can you define method body in an interface? (Java 8)**

Yes. From java 8 we can define method body in interfaces these methods are called *“Default method”.* It is also called as extension method.

Using default method we can define default implementation for an interface method

1. **If 2 interfaces have same method signature .How default method in interface is called in implemented classes? (Java 8) IMP**

Say you have 2 interfaces as below

*interface Person {*

*long getId();*

*default String getName() { return "John Q. Public"; }*

*}*

*interface Named {*

*default String getName() { return getClass().getName() + "\_" + hashCode(); }*

*}*

Both interfaces have default method with same signature. If a class implements both of these interfaces then there will be conflict that which method is called. In this case java compiler reports an error and it is up to programmer to resolve this ambiguity. In this case implementation class has to define getName method something like below based on which interface’s default method to be used

*class Student implements Person, Named {*

*public String getName() { return Person.super.getName(); }*

*...*

*}*

1. **What happens when a class implements 2 interfaces and if same method is defined as default method in one interface and as normal abstract method in another interface?**

*interface Person {*

*long getId();*

*default String getName() { return "John Q. Public"; }*

*}*

*interface Named {*

*String getName();*

*}*

* **Inheritance**

1. **What is this in java?**

It is a keyword that that refers to the current object.

1. **What is Inheritance?**

Inheritance is a mechanism in which one object acquires all the properties and behaviour of another object of another class. It represents IS-A relationship. It is used for Code Resusability and Method Overriding

1. **Which class is the super class for every class?**

Object class.

1. **Is Object class an abstract Class?**

No

1. **If object Class is not an abstract class then how methods in the class can be there without implementation?**
2. **Why multiple Inheritance is not supported in java?**

To reduce the complexity and simplify the language also multiple inheritance is rarely used.

There will be ambiguity if multiple Inheritance is supported like Say we have class A and class B both have method with same signature. Class C extends class A and B now when the ambiguous method is invoked which method will be executed is doubt.

1. **What is composition?**
2. **What is aggregation?**
3. **What is super in java?**

It is a keyword that refers to the immediate parent class object.

1. **Can you use this () and super() both in a constructor?**

No. Because super () or this() must be the first statement.

1. **What is object cloning?**

The object cloning is used to create the exact copy of an object.

1. **Parent does not have a method but using reference to call child method. What will happen in that case? IMP**
2. **What will happen if public data member has same name in base and derived class? IMP**

Method will be from overridden class variable from the object reference

* **Polymorphism**

**Method Overloading / compile time polymorphism**

1. **What is method overloading?**

If a class have multiple methods by same name but different number or type parameters, it is known as Method Overloading.

It increases the readability of the program.

1. **Why method overloading is not possible by changing the return type in java?**

Because of Ambiguity.

There will be compilation error if there are 2 methods with same name and same number and type of the argument even though the different return type

1. **Can we overload main() method?**

Yes, you can have many main () methods in a class by overloading the main method.

1. **Two methods same name... one object and other string argument if we pass null then which one will be called? IMP**

**Method Overriding or runtime polymorphism or dynamic method dispatch**

1. **What is method overriding?**

Subclass or child class to provide a specific implementation of a method that is already provided by one of its super classes or parent classes. The implementation in the subclass overrides (replaces) the implementation in the super class by providing a method that has same name, same parameters or signature, and same return type as the method in the parent class.

1. **Can we override static method?**

No, you can't override the static method because they are the part of class not object.

1. **Can we override the overloaded method?**

Yes

1. **Static binding and Dynamic binding**? **IMP**

First of all let me tell you what is binding.

Association of method definition to the method call is known as binding.

There are 2 types of binding

**Static Binding / Early Binding**

Binding resolved at compile time by compiler is known as static or early binding.

All the private, static and final methods have always been bonded at compile-time.

**Why binding of Static, final and private methods is always a static binding?**

Compiler knows that all such methods cannot be overridden and will always be accessed by object of local class.

**Dynamic Binding or Late Binding**

When compiler is not able to resolve the call/binding at compile time, such binding is known as Dynamic or late binding.

Overriding is a perfect example of dynamic binding as in overriding both parent and child classes have same method. Thus while calling the overridden method, the compiler gets confused between parent and child class method (since both the methods have same name).

Static Binding vs Dynamic Binding

**Let’s discuss the difference between static and dynamic binding in Java.**

Static binding happens at compile-time while dynamic binding happens at runtime.

Binding of private, static and final methods always happen at compile time since these methods cannot be overridden. Binding of overridden methods happen at runtime.

Java uses static binding for overloaded methods and dynamic binding for overridden methods.

http://beginnersbook.com/2013/04/java-static-dynamic-binding/

1. **Difference between method Overloading and Overriding? IMP**

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| Method with same name but different number or type of Arguments | Method with same name and same number or type of argument. |
| Occurs with a class | Occurs in 2 classes that IS-A relationship |
| Uses static binding | Uses dynamic binding |

1. **What is Runtime Polymorphism?**

An overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable. Call to the overridden method is resolved during runtime not at compile time.

1. **Can you achieve Runtime Polymorphism by data members?**

No.

1. **What is “Virtual Function”?**

Function or method whose behaviour can be overridden within an inheriting class by a function with the same signature.

Yes, all functions in Java are Virtual by default.

1. **What is covariant return type?**

Covariant return, means that when one overrides a method, the return type of the overriding method is allowed to be a subtype of the overridden method's return type

1. **Is polymorphism applies to variable?**

No.

In Java fields/member variable are not polymorphic only methods are polymorphic.

* **Final**

1. **What is final variable?**

If you make any variable as final, you cannot change the value of final variable.

1. **What is final method?**

Final methods can't be overridden.

1. **What is final class?**

Final class can't be inherited.

1. **What is blank final variable?**

A final variable, not initialized at the time of declaration, is known as blank final variable.

1. **Can we intialize blank final variable?**

Yes, only in constructor if it is non-static. If it is static blank final variable, it can be initialized only in the static block.

1. **Can you declare the main method as final?**

Yes, such as, public static final void main(String[] args){}.

* **Package**

1. **What is package?**

A package is a group of similar type of classes interfaces and sub-packages. It provides access protection and removes naming collision.

1. **Do I need to import java.lang package any time? Why?**

No. It is by default loaded internally by the JVM.

1. **Can I import same package/class twice? Will the JVM load the package twice at runtime?**

One can import the same package or same class multiple times. Neither compiler nor JVM complains about it.But the JVM will internally load the class only once no matter how many times you import the same class.

1. **What is static import?**

By static import, we can access the static members of a class directly, there is no to qualify it with the class name.

* **Nested classes**

1. **What is nested class?**

A class which is declared inside another class is known as nested class. There are 4 types of nested class member inner class, local inner class, annonymous inner class and static nested class.

1. **Is there any difference between nested classes and inner classes?**

Yes, inner classes are non-static nested classes i.e. inner classes are the part of nested classes.

1. **Can we access the non-final local variable, inside the local inner class?**

No, local variable must be constant if you want to access it in local inner class.

* **Interfaces**

1. **What is nested interface ?**

Any interface i.e. declared inside the interface or class, is known as nested interface. It is static by default**.**

1. **Can a class have an interface?**

Yes, it is known as nested interface.

1. **Can an Interface have a class?**

Yes, they are static implicitly.

1. **Why we need to have interfaces than classes? IMP**

* **Misc:**

1. **Coupling and Cohesion? IMP**

Coupling and cohesion are mainly related to quality of OO design.

A good OO design should have loose coupling and high cohesion

**Coupling:** Degree to which one class know about other class.

Say we have class A and class B. If class A knows more about class B than should know about its implementation then the classes class and class be are tightly coupled.

Say we have class which is required to read something from database using class B. In this case class A should know only getAllRecordsMethod. It should not know get connection and other processing.

**Cohesion**: Degree to which class has a single and well focused responsibility.

**Say we have Class A which is meant for database access related things. This class should not do writing to file system operation.**

1. **Different ways to create java object? IMP**
2. **Using new keyword:** Using this we can call whichever constructor we want to call no arg constructor or parameterized constructor

Employee e = new Employee();

Employee e1 = new Employee(name);

1. **Using newInstance() method of Class class**: This method calls no-arg constructor to create object.

**Employee e = (Employee) Class**.forName(“com.huawei.Employee”)

**Or**

Employee e = Employee.class.newInstance();

1. **Using newInstance() method of constructor class:**

We can also call parameterized constructor or private constructor

Constructor<Employee> constructor = Employee

Employee emp = constructor.newInstance();

1. **Using a clone() method**:

Employee emp4 = (Employee) emp3.clone();

1. **Using Deserialization:**

ObjectInputStream in = new ObjectInputStream()

Employee emp5 = (Employee) in.readObject();

Search question with dzone key to check example and more details about this feature

1. **What are wrapper classes?**

Wrapper classes are classes that allow primitive types to be accessed as objects.

1. W**hat is a native method?**

A native method is a method that is implemented in a language other than Java.

1. **What is the purpose of the System class?**

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

1. **What comes to mind when someone mentions a shallow copy in Java?**

Object cloning

1. **What is singleton class?**

Singleton class means that any given time only one instance of the class is present, in one JVM.