* Data hiding:
* Outside person can’t access our internal data directly or our internal data should not go out directly these OOP feature is nothing but data hiding.
* After validation outside person can access our internal data
* Encapsulation
* The process of binding data and corresponding methods into a single unit is nothing but it’s encapsulation
* A class is set said to be tightly encapsulated if only if each and every variable declare as private
* Has a Relationship
* Has a relationship is also known as composition and aggregation
* There is no specific keyword has a relation but most of the time we are depending on new keyword
* Main advantage of has a relationship is reusability of the code
* Without existing container object there is no chance of contained object then container and content object are strongly associated and this strong association is nothing but composition
* Without existing container object if there is chance of contained object then container and contained objects are weakly associated and this weak association is aggregiation
* If we want a total functionality of class we should go for is-a relationship
* If we want particular method then we go for has-a relationship
* Method signature
* Method signature consist of method name followed by arguments type
* Overriding
* While overriding we cannot reduce access but we can increase the scope
* Default to public but public to default we cannot
* Return type can be parent to child but not child to parent
* If child class method throws any checked exception compulsory parent class method should throws the same checked exception or parent otherwise we’ll get error but there are no restriction for unchecked exception
* We can’t override static method difference is class level method and object level method
* If both parent and child class methods are static then we won’t get any compile time error it seems overriding concept applicable for static method but it is not overriding and it’s method hiding.
* Method hiding all rules of method overriding are same .difference following
* Method hiding method should be static in overriding non stock

|  |  |
| --- | --- |
| Method Hiding | Method Overriding |
| Method should be static | Method should be non-static |
| Compile time, static polymorphism, early binding | Runtime, Dynamic, late binding polymorphism |
| Both method copy are available | Only overriding method is available |

|  |  |  |
| --- | --- | --- |
| Property | Overloading | Overriding |
| Method name | Must be same | Must be same |
| Method arguments | Must be different | Must be same |
| Method signature | Must be different | Must be same |
| Return type | No restriction | Must be covariant |
| Private static final | Can be overload | Cannot be override |
| Access Modifiers | No restriction | Access cannot be reduce can be increase |
| Throws class | No restriction | If child class method throws ant checked exception compulsory parent class should throw the same check exception or parent but no restriction for unchecked exception |
| Method resolution | Always takes care by compiler | Takes care by jvm based on runtime object |
| It is also known as | Compile time polymorphism, static polymorphism | Runtime polymorphism, dynamic ,late binding |

* Polymorphism :
* One name but multiple forms is the concept of polymorphism
* Overloading or overriding
* Usage of parent reference to hold child object is the concept of polymorphism
* If we don’t know exact runtime type of object then we should go for parent reference
* First element present in Array-list can be any-type it may be student ,customer object ,String object hence the return type of get method is object, which can hold any object
* Coupling
* The degree(measurement) of dependency between the component is called coupling
* If dependency is more the it’s consider as tightly coupling
* If dependency is less then it’s consider as loosely coupling
* Tight Coupling
* Enhancement difficulty
* Maintains will be down
* Cohesion
* For every component a clear well define functionality is defined then that component is said to be follow high cohesion
* Type casting:
* We can use parent reference to hold child object
* We can use interface reference to hold implemented class object [ Runnable r= new Thread(); ]
* A B = (C) D;
* A = class, b=reference variable, (c) = class, d=reference variable

1. Compile time checking one the type of ‘D’ and ‘C’ must have some relation either child to parent or parent to child or same type otherwise compile Exception: inconvertible types found
2. ‘C’ must be same with ‘A’ or ‘A’ must be parent otherwise compile Exception: incomputable type
3. Runtime object type of ‘D’ must be either same or derived type of c otherwise runtime exception ‘classcastexception’

* Static flow

1. identifies of static member from top to bottom
2. execution of static variable assignments and static block from top to bottom
3. execution of main method
4. at the time of class loading if we want to perform any activity when we should go for static

* Constructor

1. Constructor cannot be overriding .but can be overloaded
2. Every class in java including abstract class can content constructor but interface cannot contain
3. Recursive constructor invocation stackoverflow

Equel() and hashcode method

If two objects are equal by equal method then their hashcode must be equal that is two equivalent object should have same hashcode and they place in same bucket .

If two objects are not equel by equeal method then there is no restriction on hashcodes may be equel or may not be equel