**Day 17**

**Topic: Inheritance**

Inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such a way, you can reuse, extend or modify the attributes and behaviors which are defined in other classes.

In Java, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

**Types of Inheritance :**

**1. Single inheritance** : When one class inherits another class, it is known as single level inheritance

class Shape {

public void area() {

System.out.println("Displays Area of Shape");

}

}

class Triangle extends Shape {

public void area(int h, int b) {

System.out.println((1/2)\*b\*h);

}

}

**2. Hierarchical inheritance** : Hierarchical inheritance is defined as the process of deriving more than one class from a base class.

class Shape {

public void area() {

System.out.println("Displays Area of Shape");

}

}

class Triangle extends Shape {

public void area(int h, int b) {

System.out.println((1/2)\*b\*h);

}

}

class Circle extends Shape {

public void area(int r) {

System.out.println((3.14)\*r\*r);

}

}

**3. Multilevel inheritance** : Multilevel inheritance is a process of deriving a class from another derived class.

class Shape {

public void area() {

System.out.println("Displays Area of Shape");

}

}

class Triangle extends Shape {

public void area(int h, int b) {

System.out.println((1/2)\*b\*h);

}

}

class EquilateralTriangle extends Triangle {

int side;

}

**4. Hybrid inheritance :** Hybrid inheritance is a combination of

simple, multiple inheritance and hierarchical inheritance.

**Topic:Polymorphism**

Polymorphism is the ability to present the same interface for differing underlying forms (data types). With polymorphism, each of these classes will have different underlying data. Precisely, Poly means ‘many’ and morphism means ‘forms’.

**Runtime Polymorphism** : Runtime polymorphism is also known as dynamic polymorphism. Function overriding is an example of runtime polymorphism. Function overriding means when the child class contains the method which is already present in the parent class. Hence, the child class overrides the method of the parent class. In case of function overriding, parent and child classes both contain the same function with a different definition. The call to the function is determined at runtime is known as runtime polymorphism.

class Shape {

public void area() {

System.out.println("Displays Area of Shape");

}

}

class Triangle extends Shape {

public void area(int h, int b) {

System.out.println((1/2)\*b\*h);

}

}

class Circle extends Shape {

public void area(int r) {

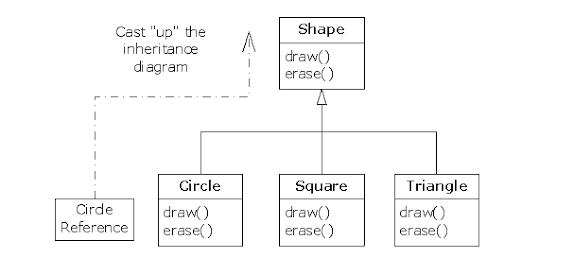
System.out.println((3.14)\*r\*r);

}

}

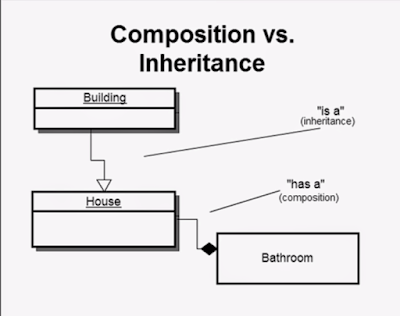
**Questions**

**Question 1: What is Inheritance in Java?**([detailed answer](http://java67.blogspot.com/2012/08/what-is-inheritance-in-java-oops-programming-example.html))  
Answer: Inheritance is an Object oriented feature which allows a class to inherit behavior and data from other class. For example, a class Car can extend basic feature of Vehicle class by using Inheritance. One of the most intuitive examples of Inheritance in the real world is Father-Son relationship, where Son inherit Father's property. If you don't know, Inheritance is the quick way to become rich :)  
  
  
**Question 2: What are different types of Inheritance supported by Java?**(detailed answer)  
Answer: Java supports single Inheritance, multi-level inheritance and at some extent multiple inheritances because Java allows a class to only extend another class, but an interface in Java can extend multiple inheritances.

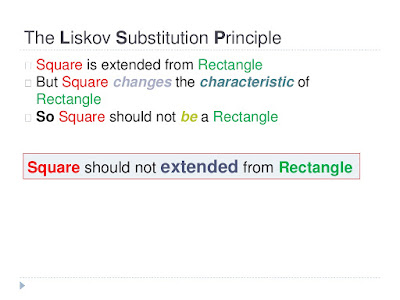
[](https://www.java67.com/2018/02/5-free-object-oriented-programming-online-courses.html)

**Question 3: Why multiple Inheritance is not supported by Java?**([detailed answer](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html))  
Answer: Java is introduced after C++ and Java designer didn't want to take some C++ feature which is confusing and not essential. They think multiple inheritances is one of them which doesn't justify complexity and confusion it introduces. You can also check why multiple inheritances are not supported in Java for more reasons and discussion around this.  
  
  
**Question 4: Why Inheritance is used by Java Programmers?**([detailed answer](http://java67.blogspot.com/2015/12/top-30-oops-concept-interview-questions-answers-java.html))  
Answer: Inheritance is used for code reuse and leveraging Polymorphism by creating a type hierarchy. It's better to use Inheritance for type declaration but for code reuse composition is a better option because it's more flexible. See this article for learning more about why Composition is better than Inheritance.  
  
  
**Question 5: How to use Inheritance in Java?**([detailed answer](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html))  
Answer: You can use Inheritance in Java by extending classes and implementing interfaces. Java provides two keywords extends and implements to achieve inheritance.  A class which is derived from another class is known as a subclass and an interface which is derived from another interface is called subinterface. A class which implements an interface is known as implementation.  
  
  
 **Question 6: What is the syntax of Inheritance?**(detailed answer)  
Answer: You can use either extends of implements keyword to implement Inheritance in Java.  A class extends another class using extends keyword, an interface can extend another interface using extend keyword, and a class can implement an interface using implements keyword in Java.  
  
  
**Question 7: What is the difference between Inheritance and Encapsulation?**(detailed answer)  
Answer: Inheritance is an object oriented concept which creates a parent-child relationship. It is one of the ways to reuse the code written for parent class but it also forms the basis of Polymorphism. On the other hand, Encapsulation is an object oriented concept which is used to hide the internal details of a class e.g. HashMap encapsulate how to store elements and how to calculate hash values.  
  
  
**Question 8: What is the difference between Inheritance and Abstraction?**(detailed answer)  
Answer: Abstraction is an object oriented concept which is used to simply things by abstracting details. It helps in the designing system. On the other hand, Inheritance allows code reuse. You can reuse the functionality you have already coded by using Inheritance. See [Head First Object Oriented Analysis and Design](https://www.java67.com/2016/10/top-5-object-oriented-analysis-and-design-patterns-book-java.html) for more details.

**Question 9: What is the difference between Polymorphism and Inheritance?** ([detailed answer](http://java67.blogspot.com/2014/04/difference-between-polymorphism-and-Inheritance-java-oops.html))  
Answer: Both Polymorphism and Inheritance goes hand on hand, they help each other to achieve their goal. Polymorphism allows flexibility, you can choose which code to run at runtime by overriding.  See the detailed answer for more details.  
  
 **Question 10: What is the difference between Composition and Inheritance in OOP?**([detailed answer](http://javarevisited.blogspot.com/2015/06/difference-between-inheritance-and-Composition-in-Java-OOP.html))  
Answer: One of the good question to check the candidate's object-oriented programming skills. There are several differences between Composition and Inheritance in Java, some of them are following:  
  
1. The Composition is more flexible because you can change the implementation at runtime by calling setXXX() method, but Inheritance cannot be changed i.e. you cannot ask a class to implement another class at runtime.  
  
2. Composition builds HAS-A relationship while Inheritance builds IS-A relationship e.g. A Room HAS A Fan, but Mango IS-A Fruit.  
  
3. The parent-child relationship is best represented using Inheritance but If you just want to use the services of another class use Composition. For more differences see [5 reasons to favor composition over Inheritance](http://javarevisited.blogspot.com/2013/06/why-favor-composition-over-inheritance-java-oops-design.html).

[](https://javarevisited.blogspot.com/2018/02/top-5-java-design-pattern-courses-for-developers.html)

**11. Can we override static method in Java?**([detailed answer](http://java67.blogspot.com/2012/08/can-we-override-static-method-in-java.html))  
No, you cannot override a static method in Java because it's resolved at compile time. In order for overriding to work, a method should be virtual and resolved at runtime because objects are only available at runtime. This is one of the tricky Java questions, where interviewer tries to confuse you. A programmer is never sure about whether they can override or overload a static method in Java.  
  
  
**12. Can we overload a static method in Java?**([detailed answer](http://java67.blogspot.com/2012/08/can-we-overload-static-method-in-java.html))  
Yes, you can overload a static method in Java. Overloading has nothing to do with runtime but the signature of each method must be different. In Java, to change the method signature, you must change either number of arguments, type of arguments or order of arguments.  
  
 **13. Can we override a private method in Java?**([detailed answer](http://java67.blogspot.com/2012/08/can-we-override-private-method-in-java.html))  
No,  you cannot override a private method in Java because the private method is not inherited by the subclass in Java, which is essential for overriding. In fact, a private method is not visible to anyone outside the class and, more importantly, a call to the private method is resolved at compile time by using Type information as opposed to runtime by using the actual object.  
  
 **Question 14: What is method hiding in Java?**([detailed answer](http://java67.blogspot.com/2015/08/top-10-method-overloading-overriding-interview-questions-answers-java.html))  
Answer: Since the static method cannot be overridden in Java, but if you declare the same static method in subclass then that would hide the method from the superclass. It means, if you call that method from subclass then the one in the subclass will be invoked but if you call the same method from superclass then the one in superclass will be invoked. This is known as method hiding in Java.  
  
  
**Question 15: Can a class implement more than one interface in Java?**([detailed answer](http://java67.blogspot.com/2014/02/what-is-actual-use-of-interface-in-java.html))  
Yes, A class can implement more than one interface in Java e.g. A class can be both Comparable and Serializable at the same time. This is why the interface should be the best use for defining Type as described in Effective Java. This feature allows one class to play a polymorphic role in the program.  
 **Question 16: Can a class extends more than one class in Java?**(detailed answer)  
Answer: No, a class can only extend just one more class in Java.  Though Every class also, by default extend the java.lang.Object class in Java.  
  
  
**Question 17: Can an interface extends more than one interface in Java?**([answer](http://java67.blogspot.com/2013/07/15-advanced-core-java-interview-questions-answers-senior-experienced-5-6-years-programmers-developers.html))  
Answer: Yes, unlike classes, an interface can extend more than one interface in Java. There are several example of this behavior in JDK itself e.g. java.util.List interface extends both Collection and Iterable interface to tell that it is a Collection as well as it allows iteration via Iterator.  
  
  
**18: What will happen if a class extends two interfaces and they both have a method with same name and signature?**(detailed answer)  
In this case, a conflict will arise because the compiler will not able to link a method call due to ambiguity. You will get a compile time error in Java.  
  
 **Question 19: Can we pass an object of a subclass to a method expecting an object of the super class?**([answer](http://java67.blogspot.com/2013/07/15-advanced-core-java-interview-questions-answers-senior-experienced-5-6-years-programmers-developers.html))  
Answer: Yes, you can pass that because subclass and superclass are related to each other by Inheritance which provides IS-A property.  I mean Banana is a Fruit so you can pass banana if somebody expect fruit. Now there are scenario, where you can't do e.g. when subclass violates the Liskov Substitution principle i.e. you cannot pass a plastic banana to someone expecting fruit :-), The eat() function will throw exception.

[](https://javarevisited.blogspot.com/2018/07/10-object-oriented-design-principles.html)

**Question 20: What is the Liskov substitution principle?**(detailed answer)  
Answer: The Liskov substitution principle is one of the five object-oriented design principles, collectively know as [SOLID principles](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html).

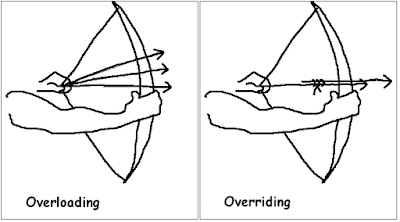
This design principle is L of SOLID acronym. The Liskov substitution principle states that in an object oriented program if a function or method is expecting an object of base class then  it should work fine with a derived class object as well. If it cannot function properly with derived class object then the derived class is violating the Liskov Substitution principle.  
  
For example, if a method is expecting a List you can also pass [ArrayList or LinkedList](http://java67.blogspot.com/2012/12/difference-between-arraylist-vs-LinkedList-java.html" \t "_blank) and it should work just fine because ArrayList and LinkedList both follow Liskov Substitution Principle, but the [java.sql.Date](http://java67.blogspot.com/2014/02/how-to-convert-javautildate-to-javasqldate-example.html) which is a subclass of java.util.Date in Java violates Liskov Substitution Principle because you cannot pass an object of java.sql.Date class to a method which is expecting an object of java.util.Date, Why? because all time-related method will throw java.lang.UnsupportedOperationException.

**1) What is method overloading in Java?**  
If you have two methods which do the same thing it's better they have the same name, but two methods cannot have the same name until you overload them. So overloading is a process of declaring two methods with the same name but different method signature like System.out which is object of PrintStream class has several println() method to print different data types e.g. byte, short, int, char, float and double. All of them are called the overloaded method. Overloaded method calls are resolved during compile time in Java and they must have different method signatures. See [here](http://java67.blogspot.sg/2012/08/what-is-method-overloading-in-java-example.html) to learn more about method overloading in Java.  
  
  
**2) What is method overriding in Java?**  
Method overriding is another way to define a method with same name but different code but it must be in sub class. Overriding is based upon run-time Polymorphism as method calls are resolved at run-time depending upon actual object.  For example if a variable of type Parent holds an object of Child class then method invoked will be from child class and not parent class, provides its overridden. In order to override a method, you must follow rules of method overriding which means declaring method with same signature in sub class. See [here](http://java67.blogspot.sg/2012/08/what-is-method-overriding-in-java-example-tutorial.html) to learn more about method overriding in Java.  
  
  
**3) What is method hiding in Java?**  
static method cannot be overriding in Java because their method calls are resolved at compile time but it didn't prevent you from declaring method with same name in sub class. In this case we say that method in sub class has hidden static method from parent class. If you have a case where variable of Parent class is pointing to object of Child class then also static method from Parent class is called because overloading is resolved at compile time. See [here](http://java67.blogspot.sg/2012/08/can-we-override-static-method-in-java.html) to learn more about method hiding in Java.  
  
  
**4) What are the rules of overloading a method in Java?**  
One and only rule of method overloading in Java is that the method signature of all overloaded method must be different. Method signature is changed by changing either number of method arguments, or type of method arguments e.g. System.out.println() method is overloaded to accept different primitive types like int, short, byte, float etc.

They all accept just one argument but their type is different. You can also change method signature by changing order of method argument but that often leads to ambiguous code so better to be avoided. See [here](http://java67.blogspot.sg/2012/09/what-is-rules-of-overloading-and-overriding-in-java.html) for full list of rules.  
  
  
**5) Difference between method overloading and overriding?**  
The fundamental difference between overloading and overriding is that formerly took place during compile time while later took place during run-time.

Due to this reason, its only possible to overload virtual methods in Java. You cannot overload methods which are resolved during compile time e.g. private, static and final method cannot be overridden in Java.

Also, rules of method overloading and overriding are different, for example in order to overload a method its method signature must be different but for the overriding method, it must be the same. See this image to learn more difference between overriding and overloading in Java.

[](http://java67.blogspot.sg/2012/09/difference-between-overloading-vs-overriding-in-java.html)

**6) Can we overload static method in Java?**  
Yes, its possible to overload static method in Java. You can declare as many static method with same name as you want until all of them have different method signature. Remember, return type is not part of method signature, so they must have either different number of arguments, or different type of argument.

There is a third option also which changes order of argument but I suggest not to do that because it often result in the ambiguous method call. It's very important to follow [these best practices](http://javarevisited.blogspot.sg/2013/01/java-best-practices-method-overloading-constructor.html) while overloading a static method in Java.  
  
  
**7) Can we override static method in Java?**  
No, you cannot override static method in Java because they are resolved and bonded during compile time. Since overriding is a run-time activity and if a method call is already resolved at compile time then it will not take place and that's why its not possible to override static method in Java.

But, you can define another static method of same signature in sub class, this is known as method hiding. Actual method called will depends upon the type of class and not on type of object as its the case with overriding. See [here](http://javarevisited.blogspot.sg/2013/03/can-we-overload-and-override-static-method-java.html) to learn more about why you cannot override static method in Java.  
  
  
**8) Can you prevent overriding a method without using final modifier?**  
Yes, there are some funky ways to prevent method overriding in Java. Though final modifier is only for that purpose you can use private keyword to prevent method overriding. How? If you remember correctly, in order to override a method, the class must be extensible. If you make the constructor of parent class private then its not possible to extend that class because its constructor will not be accessible in sub class, which is automatically invoked by sub class constructor, hence its not possible to override any method from that class. This technique is used in Singleton design pattern, where constructor is purposefully made private and a static getInstance() method is provided to access singleton instance. See [here](http://javarevisited.blogspot.sg/2015/04/3-ways-to-prevent-method-overriding-in.html) to learn more techniques to prevent method overriding in Java.  
  
  
**9) Can we override a private method in Java?**  
No, you cannot override private method in Java. Since private methods are not visible outside the class, they are not available in sub-class hence they cannot be overridden. By the way, how about overriding a private method inside an Inner class? Is it possible? See [here](http://java67.blogspot.sg/2013/08/can-we-override-private-method-in-java-inner-class.html) to learn more why you cannot override private method in Java.  
  
  
**10) What is co-variant method overriding?**  
One of the rule of method overriding is that return type of overriding method must be same as overridden method but this restriction is relaxed little bit from Java 1.5 and now overridden method can return sub class of return type of original method. This relaxation is known as co-variant method overriding and it allows you to remove casting at client end.

One of the best examples of this comes [when you override clone() method](http://javarevisited.blogspot.sg/2015/01/java-clone-tutorial-part-2-overriding-with-mutable-field-example.html). Original Object.clone() method returns Object which needs to cast, but with co-variant method overriding you can directly return relevant type e.g. Date class returns object of java.util.Date instead of java.lang.Object. See here to learn more about co-variant method overriding in Java.  
  
  
**11) Can we change argument list of overridden method?**  
No, you cannot change the argument list of overridden method in Java. An overriding method must have same signature as original method. Only return type can be changed that to only to sub type of return type of original method.  
  
  
**12) Can we change return type of method in subclass while overriding?**  
No, you cannot change the return type of method during overriding. It would be violation of rules of overriding. Though from Java 5 onward you can replace the return type with sub type e.g. if original method has return type as java.lang.Object then you can change return type of overridden method as any type e.g. clone() method. This is also known as [co-variant method overriding](http://javarevisited.blogspot.sg/2014/03/covariant-method-overriding-of-java-5.html) in Java.  
  
  
**13) Can we override a method which throws run-time exception without throws clause?**  
Yes, you can. There is no restriction on throwing RuntimeException from overriding method. So if your original method throws NullPointerException than its not necessary to throw NPE from overriding method as well.  
  
  
**14) How do you call super class version of an overriding method in sub class?**  
You can call it using super keyword. For example if you have a method calculate() in both parent and child class then from child class you can invoke parent class method calculate() as super.calculate(). It's very similar to calling super class constructor from sub class as shown [here](http://java67.blogspot.sg/2012/12/how-constructor-chaining-works-in-java.html).  
  
  
**15) What are rules of method overriding in Java?**  
Some rules of method overriding are following :

* Overriding method cannot throw higher exception than overridden one, but that's only true for checked exception.
* Overriding method cannot restrict access of overridden method e.g. if original method is public then overriding method must be public, but it can expand access e.g. if original method is protected than overriding method can be protected or public.

See [here](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html) for full list of rules of method overriding in Java.  
  
  
**16) Can we override a non-static method as static in Java?**  
No, its not possible to define a non-static method of same name as static method in parent class, its compile time error in Java. See here to learn more about the [overriding static method in Java](http://java67.blogspot.sg/2012/11/what-is-static-class-variable-method.html).  
  
  
**17) Can we override the constructor in Java?**  
No, you cannot override constructors in Java because they are not inherited. Remember, we are talking about overriding here not overloading, you can overload construct but you cannot override them. Overriding always happens in child class and since constructors are not inherited and their name is always the same as the class name it's not possible to override them in Java, to learn more about constructors see [here](http://java67.blogspot.sg/2014/09/Why-constructor-is-important-in-java-example.html)  
  
  
**18) Can we override a final method in Java?**  
No, you cannot override a final method in Java. Trying to override a final method in a subclass will result in a compile-time error. Actually making a method final is signal to all developer that this method is not for inheritance and it should be used in its present form. You generally make a method final due to security reasons, to learn more see [here](http://javarevisited.blogspot.sg/2013/12/when-to-make-method-final-in-java.html).  
  
  
**19) Can you overload or override the main method in Java?**  
Since main() is a static method in Java, it follows rules associated with the static method, which means you can overload the main method but you cannot override it. By the way, even if you overload the main method, JVM will always call the standard public static void main(String args[]) method to start your program, if you want to call your overloaded method you need to do it explicitly in your code as shown [here](http://java67.blogspot.sg/2015/06/can-you-overload-or-override-main-in-java.html).