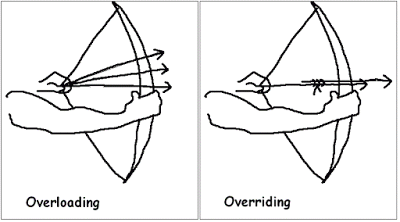
**19 Java Method Overloading and Overriding Interview Questions and Answers**

**1) What is method overloading in Java?**  
If you have two method which does same thing its better they have same name, but two method cannot have same name until you overload them. So overloading is a process of declaring two methods with same name but different method signature e.g. 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 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 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 hided 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 rules of overloading a method in Java?**  
One and only rule of method overloading in Java is that 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?**  
Fundamental difference between overloading and overriding is that former 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 overriding method it must be same. See this image to learn more difference between overriding and overloading in Java.



**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 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 example 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 [overriding static method in Java](http://java67.blogspot.sg/2012/11/what-is-static-class-variable-method.html).  
  
  
**17) Can we override constructor in Java?**  
No, you cannot override constructor 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 at child class and since constructor are not inherited and their name is always same as the class name its not possible to override them in Java, to learn more about constructor see [here](http://java67.blogspot.sg/2014/09/Why-constructor-is-important-in-java-example.html)  
  
  
**18) Can we override final method in Java?**  
No, you cannot override final method in Java. Trying to override final method in subclass will result in compile time error. Actually making a method final is signal to all developer that this method is not for inheritance and it should be use 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 main method in Java?**  
Since main() is a static method in Java, it follows rules associated with static method, which means you can overload main method but you cannot override it. By the way, even if you overload a 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).  
  
Read more: <http://www.java67.com/2015/08/top-10-method-overloading-overriding-interview-questions-answers-java.html#ixzz4g83QuQQq>