**[10 Interview Questions on Java Generics for Programmer and Developers](http://javarevisited.blogspot.ru/2012/06/10-interview-questions-on-java-generics.html" \o "10 Interview Questions on Java Generics for Programmer and Developers)**

**Generic interview questions** in [Java interviews](http://javarevisited.blogspot.sg/2011/04/top-20-core-java-interview-questions.html) are getting more and more common with Java 5 around there for considerable time and many application either moving to Java 5 and almost all new Java development happening on Tiger(code name of Java 5).  Importance of Generics and Java 5 features like [Enum](http://javarevisited.blogspot.com/2011/08/enum-in-java-example-tutorial.html),  [Autoboxing](http://javarevisited.blogspot.sg/2012/07/auto-boxing-and-unboxing-in-java-be.html), [varargs](http://javarevisited.blogspot.sg/2011/09/variable-argument-in-java5-varargs.html) and Collection utilities like [CountDownLatch](http://javarevisited.blogspot.sg/2012/07/countdownlatch-example-in-java.html), [CyclicBarrier](http://javarevisited.blogspot.sg/2012/07/cyclicbarrier-example-java-5-concurrency-tutorial.html) and [BlockingQueue](http://javarevisited.blogspot.sg/2012/02/producer-consumer-design-pattern-with.html) are getting more and more popular on Java interviews. Generic interview question can get real tricky if you are not familiar with [bounded and unbounded generic wildcards](http://javarevisited.blogspot.com/2012/04/what-is-bounded-and-unbounded-wildcards.html), **How generics works internally**, type erasure and familiarity with writing parametrized generics classes and methods in Java. Best way to prepare for Generics interview is to try simple program best on various features of generics. Anyway In this Java interview article we will see some popular Java Generics interview questions and there answer. By the way there are lot of material available in Javarevisited to better preparing for Java and J2EE interviews, you can prepare multi-threading and Collections using [15 thread interview question](http://javarevisited.blogspot.sg/2011/07/java-multi-threading-interview.html) and [Top 10 Java collection interview question](http://javarevisited.blogspot.sg/2011/11/collection-interview-questions-answers.html) along with several other questions answers articles on Spring, Struts, JSP and Servlet. If you are GUI developers and working in Java Swing technology than you can also check interview [questions on Java Swing](http://javarevisited.blogspot.sg/2011/09/swing-interview-questions-answers-in.html) mostly asked in Investment banks

**Java Generics Interview Questions**

**1. What is Generics in Java ? What are advantages of using Generics?**

This is one of the first interview questions asked on generics in any Java interview, mostly at beginners and intermediate level. Those who are coming from prior to Java 5 background knows that how inconvenient it was to [store object in Collection](http://javarevisited.blogspot.com/2012/07/create-read-only-list-map-set-example-java.html) and then cast it back to correct Type before using it. Generics prevents from those. it provides [compile time](http://javarevisited.blogspot.com/2012/03/what-is-static-and-dynamic-binding-in.html) type-safety and ensures that you only insert correct Type in collection and avoids ClassCastException in runtime.

**2. How Generics works in Java ? What is type erasure ?**

This is one of better interview question in Generics. Generics is implemented using **Type erasure**, compiler erases all type related information during compile time and no type related information is available during runtime. for example List<String> is represented by only [List](http://www.blogger.com/goog_1304192070)at runtime. This was done to ensure binary compatibility with the libraries which were developed prior to Java 5. you don't have access to Type argument at runtime and **Generic type** is translated to **Raw type** by compiler during [runtime](http://javarevisited.blogspot.sg/2012/03/what-is-static-and-dynamic-binding-in.html). you can get lot of follow up question based on this Generic interview question based upon your response e.g. *Why Generics is implemented using Type erasure* or presenting some invalid generic code which results in compiler error. read my post [How Generics works in Java](http://javarevisited.blogspot.com/2011/09/generics-java-example-tutorial.html) for more details

**3. What is Bounded and Unbounded wildcards in Generics ?**

This is another very [popular Java interview questions](http://javarevisited.blogspot.com/2011/04/top-20-core-java-interview-questions.html) on Generics. Bounded Wildcards are those which impose bound on Type. there are two kinds of Bounded wildcards <? extends T> which impose an upper bound by ensuring that type must be sub class of T and <? super T> where its imposing lower bound by ensuring Type must be super class of T. This Generic Type must be instantiated with Type within bound otherwise it will result in compilation error. On the other hand <?> represent and unbounded type because <?> can be replace with any Type. See more on my post [differences between Bounded and Unbounded wildcards in Generics](http://javarevisited.blogspot.com/2012/04/what-is-bounded-and-unbounded-wildcards.html).

**4. What is difference between List<? extends T>  and  List <? super T> ?**

This is related to previous generics interview questions, some time instead of asking what is bounded and unbounded wildcards interviewer present this question to gauge your understanding of generics. Both of List declaration is e[xample of bounded wildcards,](http://javarevisited.blogspot.sg/2012/04/what-is-bounded-and-unbounded-wildcards.html) List<? extends T> will accept any List with Type extending T while List<? super T> will accept any List with type super class of T. for Example List<? extends Number> can accept List<Integer> or List<Float>. see more on above link.

**5. How to write a generic method which accepts generic argument and return Generic Type?**

[Interview questions on Generics in Java 2 years](http://javarevisited.blogspot.sg/2011/09/swing-interview-questions-answers-in.html)writing generic method is not difficult, instead of using raw type you need to use Generic Type like T, E or K,V which are well known placeholders for Type, Element and Key, Value. Look on [Java Collection framework](http://javarevisited.blogspot.sg/2011/11/collection-interview-questions-answers.html) for examples of generics methods. In simplest form a generic method would look like:

**public** V put(K key, V value) {  
        **return** cache.put(key, value);  
}

**6. How to write parametrized class in Java using Generics ?**

This is an extension of previous Java generics interview question. Instead of asking to write Generic method Interviewer may ask to *write a type safe class using generics*. again key is instead of using raw types you need to used generic types and always use standard place holder used in JDK.

**7. Write a program to implement LRU cache using Generics ?**

This is an exercise for anyone who like [Coding in Java](http://javarevisited.blogspot.sg/2011/09/code-review-checklist-best-practice.html). One hint is that LinkedHashMap can be used implement fixed size LRU cache  where one needs to remove eldest entry when Cache is full. LinkedHashMap provides a method called removeEldestEntry() which is called by put() and putAll() and can be used to instruct to remove eldest entry. you are free to come up with your own implementation as long as you have a written a working version along with [JUnit test](http://javarevisited.blogspot.sg/2012/06/junit4-annotations-test-examples-and.html).

**8. Can you pass List<String> to a method which accepts List<Object>**

This generic interview question in Java may look confusing to any one who is not very familiar with Generics as in fist glance it looks like String is object so List<String> can be used where List<Object> is required but this is not true. It will result in compilation error. It does make sense if you go one step further because List<Object> can store any any thing including [String, Integer](http://javarevisited.blogspot.com/2011/08/convert-string-to-integer-to-string.html) etc but List<String> can only store Strings.

**List**<**Object**> objectList;  
**List**<**String**> stringList;  
        
objectList = stringList;  *//compilation error incompatible types*

**9. Can we use Generics with Array?**

This was probably most simple generics interview question in Java, if you know the fact that Array doesn't support Generics and that's why Joshua Bloch suggested in [Effective Java](http://www.amazon.com/dp/0321356683/?tag=javamysqlanta-20) to prefer List over Array because *List can provide compile time type-safety* over Array.

**10. How can you suppress unchecked warning in Java ?**

javac compiler for Java 5 generates unchecked warnings if you use combine raw types and generics types e.g.

**List**<**String**> rawList = **new** **ArrayList**()  
Note: Hello.java uses unchecked or unsafe operations.;

which can be suppressed by using @SuppressWarnings("unchecked") annotation.

**Java Generics Interview questions Update:**

I got few more interview questions on Generics in Java to share with you guys, These questions focus on What is difference between Generics type and Raw type and Can we use Object in place of bounded wildcards etc:

**Difference between List<Object> and raw type List in Java?**

Main difference between raw type and parametrized type List<Object> is that, [compiler](http://javarevisited.blogspot.sg/2011/12/jre-jvm-jdk-jit-in-java-programming.html) will not check type-safety of raw type at compile time but it will do that for parametrized type and by using Object as Type it inform compiler that it can hold any Type of Object e.g. String or Integer. This Java Generics interview question is based on correct understanding of raw type in Generics. Any way second difference between them is that you can pass any parametrized type to raw type List but you can not pass List<String> to any method which accept List<Object> it will result in compilation error. Read [How Generics works in Java](http://javarevisited.blogspot.sg/2011/09/generics-java-example-tutorial.html) for more details.

**Difference between List<?> and List<Object> in Java?**

This generics interview question may look related to previous interview questions but completely different. *List<?> is List of unknown type* while *List<Object> is essentially List of any Type*. You can assign List<String>, List<Integer> to List<?> but you can not assign List<String> to List<Object>.

**List**<?> listOfAnyType;  
**List**<**Object**> listOfObject = **new** **ArrayList**<**Object**>();  
**List**<**String**> listOfString = **new** **ArrayList**<**String**>();  
**List**<**Integer**> listOfInteger = **new** **ArrayList**<**Integer**>();  
        
listOfAnyType = listOfString; *//legal*  
listOfAnyType = listOfInteger; *//legal*  
listOfObjectType = (**List**<**Object**>) listOfString; *//compiler error - in-convertible types*

to know more about wildcards see [Generics Wildcards Examples in Java](http://javarevisited.blogspot.sg/2011/09/generics-java-example-tutorial.html)

**Difference between List<String> and raw type List.**

This Generics interview question is similar to difference between raw type and parametrized type.Parametrized type are type-safe and type-safety will be guaranteed by compiler but [List raw type is not type safe](http://javarevisited.blogspot.sg/2012/04/difference-between-list-and-set-in-java.html). You can not store any other Object on List of String but you can not store any Object in raw List. There is no casting required in case of Parametrized type with Generics but explicit casting will be needed for raw type.

**List** listOfRawTypes = **new** **ArrayList**();  
listOfRawTypes.add("abc");  
listOfRawTypes.add(123); *//compiler will allow this - exception at runtime*  
**String** item = (**String**) listOfRawTypes.get(0); *//explicit cast is required*  
item = (**String**) listOfRawTypes.get(1); *//ClassCastException because Integer can not be cast in String*  
        
**List**<**String**> listOfString = **new** **ArrayList**();  
listOfString.add("abcd");  
listOfString.add(1234); *//compiler error, better than runtime Exception*  
item = listOfString.get(0); *//no explicit casting is required - compiler auto cast*

These were some of the *frequently asked generics interview questions and answers in Java*. None of these generic interview questions are tough or hard, Indeed they are based on fundamental knowledge of generics. Any Java programmer who has decent knowledge of Generics must be familiar with these generics questions in Java. If you have any other good generic question which has been asked in any interview or you are looking answer for any Generics interview question in Java then please post in comment section.

Read more: <http://javarevisited.blogspot.com/2012/06/10-interview-questions-on-java-generics.html#ixzz2lbRy4uMY>