Generics
Assignment

Q1. What are Generics in java?

Ans. The Java Generics programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time.

Before generics, we can store any type of objects in the collection, i.e., non-generic. Now generics force the java programmer to store a specific type of objects.

Q2. What are the benefits of using generics in java?

Ans. Type safety:

One of the most significant benefits of using generics in Java is that it enhances the type safety of your code. With generics, you can specify the type of data that a class, method, or interface can work with. This ensures that only the specified data types are passed to the code, preventing runtime errors and improving the reliability of your program.

Code reuse:

Generics in Java allow you to write code that is more reusable. By specifying a type parameter in a generic class, method, or interface, you can create code that can work with multiple data types. This reduces code duplication, improves code maintainability, and makes your code more efficient.

Compile-time checking:

Another significant advantage of using generics in Java is that it enables compile-time checking of your code. This means that errors can be detected early in the development process, rather than at runtime. This not only improves the reliability of your program but also saves time and effort in debugging.

Improved performance:

Generics in Java can also improve the performance of your program. By using generics, the compiler can optimise your code, reducing the number of casts and improving the overall performance of your program.

Greater flexibility:

Generics in Java provide greater flexibility in your programming. By allowing you to specify the type of data that a class, method, or interface can work with, generics make it possible to create highly adaptable and flexible code. This can be especially useful when working with complex data structures or when you need to write code that can work with multiple data types.

Better documentation:

Finally, generics in Java can improve the documentation of your code. By specifying the type parameter of a generic class, method, or interface, you can make it clear to other developers what data types the code can work with. This makes your code more readable, easier to understand, and more maintainable.

Avoiding casting:

Generics in Java allow you to avoid casting, which is a common source of bugs and errors in Java programs. By specifying the type of data that a class, method, or interface can work with, generics eliminate the need for casting, making your code more concise, readable, and maintainable.

Easier debugging:

Generics in Java can also make debugging easier. By providing more information about the data types that your code can work with, generics can help you identify errors and bugs more quickly and accurately. This can save you time and effort in the debugging process, improving the overall quality of your code.

Interoperability:

Generics in Java also make it easier to write code that can work with other Java libraries and frameworks. By using generics, you can ensure that your code is compatible with other Java code that uses generics, making it easier to integrate your code with other software and systems.

Q3. What is a Generic Class in java?

Ans. Java Generics was introduced to deal with type-safe objects. It makes the code stable.Java Generics methods and classes, enables programmer with a single method declaration, a set of related methods, a set of related types. Generics also provide compile-time type safety which allows programmers to catch invalid types at compile time. Generic means parameterized types. Using generics, the idea is to allow any data type to be Integer, String, or any user-defined Data Type and it is possible to create classes that work with different data types.

A Generic class simply means that the items or functions in that class can be generalised with the parameter(example T) to specify that we can add any type as a parameter in place of T like Integer, Character, String, Double or any other user-defined type.

Syntax -:

```
class Solution<T>
{
```

```
    T data;
    public static T getData(){
        return data;
    }
}
```

Example-:

```
public class Area<T> {

        private T t;

     public void add(T t)
    {
               this.t = t;
    }

     public T get() { return t; }

     public void getArea() {}

     public static void main(String[] args)
    {

         Area<Integer> rectangle = new Area<Integer>();
        Area<Double> circle = new Area<Double>();
        rectangle.add(10);
        circle.add(2.5);
        System.out.println(rectangle.get());
        System.out.println(circle.get());
    }
}
```

Q4. What is a Type Parameter in java generics?

Ans. The type parameter section, delimited by angle brackets (<>), follows the class name. It specifies the type parameters (also called type variables) T1, T2, ..., and Tn.

To update the Box class to use generics, you create a generic type declaration by changing the code "public class Box" to "public class Box<T>".

Q5. What is a generic method in java?

Ans . The angle bracket syntax is required to declare a type parameter when writing a generic method in Java. A placeholder for the real type that will be supplied to the method at runtime is the type argument.
Example-:

```
1. public <T> void methodName(T parameter) {
2.     // method implementation
3. }
```

Example-:

```
1. public class GenericMethod1{
2.     public static void main(String[] args) {
3.         Integer[] intArray = { 1, 2, 3, 4, 5 };
4.         String[] stringArray = { "Hello", "World", "!" };
5.         System.out.print("Integer Array: ");
6.         printArray(intArray);
7.         System.out.print("String Array: ");
8.         printArray(stringArray);
9.     }
10.    public static <T> void printArray(T[] arr) {
11.        for (T element : arr) {
12.            System.out.print(element + " ");
13.        }
14.        System.out.println();
15.    }
16.}
```

Q6. What is the difference between ArrayList and ArrayList<T>?

Ans. `ArrayList<` in Java`?>`, which was actually asked to him in a recent Java development interview. The key difference between them is that

`ArrayList` is not using Generics while `ArrayList` is a generic ArrayList but they look very similar. If a method accepts `ArrayList` or `ArrayList<?>` as a parameter then it can accept any type of `ArrayList` like ArrayList of `String`, `Integer`, `Date`, or `Object`, but if you look closely you will find that one is raw type while the other is using an unbounded wildcard. What difference that could make? Well, that makes a significant difference because `ArrayList` with raw type is not type-safe but `ArrayList<?>` with the **unbounded wildcard is type-safe.**