**Generics**

**Assignment**

1. What are Generics in Java?

Ans: Generics in Java are used to provide type safety and reduce code redundancy by allowing the use of generic types. It allows classes, methods, and interfaces to be written generically, without specifying the type of data being used.

1. What are the benefits of using Generics in Java?

Ans: The benefits of using Generics in Java are:

* + - Type safety
    - Code reusability
    - Improved readability
    - Reduced code redundancy
    - Improved performance

1. What is a Generic Class in Java?

Ans:  A generic class is implemented exactly like a non-generic class. The only difference is that it contains a type parameter section. There can be more than one type of parameter, separated by a comma. The classes, which accept one or more parameters, ​are known as parametrized classes or parameterized types.

Example:

class container<T>

{

private T obj1;

public void add(T obj1)

{

this.obj1 = obj1;

}

public T get()

{

return obj1;

}

public static void main(String[] args)

{

container<Integer> integerContainer= new container<Integer>();

container<String> stringContainer = new container<String>();

integerContainer.add(new Integer(7));

stringContainer.add(new String("You are awesome"));

System.out.printf("Integer Value :%d\n\n", integerContainer.get());

System.out.printf("String Value :%s\n", stringContainer.get());

}

}

//Output:

// Integer Value: 7

// String Value : You are awesome

1. What is a Type Parameter in Java Generics?

Ans:   A Type Parameter in Java Generics is a placeholder for the type of data that is used by a generic class or method. It is defined using a single uppercase letter enclosed in angle brackets, such as <T> or <E>.

1. What is a Generic Method in Java?

Ans: Generic Java method takes a parameter and returns some value after performing a task. It is exactly like a normal function, however, a generic method has type parameters which are cited by actual type. This allows the generic method to be used in a more general way. The compiler takes care of the type of safety which enables programmers to code easily since they do not have to perform long, individual type castings.

Example:

class GenericMethodTest

{

// generic method printArray

public static < E > void printArray( E[] inputArray )

{

// Display array elements

for(E element : inputArray)

{

System.out.printf("%s ", element);

}

System.out.println();

}

public static void main(String args[])

{

// Create arrays of Integer, Double and Character

Integer[] integerArray = { 5, 4, 3, 2, 1 };

Double[] doubleArray = { 1.21, 22.12, 13.32 };

Character[] characterArray = { 'Y', 'o', 'u', ' ', 'a', 'r', 'e', ' ', 'a','w','e','s','o','m','e' };

System.out.println("integerArray contains:");

printArray(integerArray); // pass an Integer array

System.out.println("\ndoubleArray contains:");

printArray(doubleArray); // pass a Double array

System.out.println("\ncharacterArray contains:");

printArray(characterArray); // pass a Character array

}

}

//Output:

// integerArray contains:

// 5 4 3 2 1

// doubleArray contains:

// 1.21 22.12 13.32

// characterArray contains:

// y o u a r e a w e s o m e

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

Ans: ArrayList is a non-generic class, while ArrayList<T> is a generic class. ArrayList<T> provides type safety, as it can only store elements of the specified type, whereas ArrayList can store any type of element.

**IO Operations**

**Assignment**

1. What is Input and Output Stream in Java?

Ans: **Input Stream-**The InputStream class of the java.io package is an abstract superclass that represents an input stream of bytes. InputStream is an abstract class, it is not useful by itself. However, its subclasses can be used to read data.

**Output Stream-**The OutputStream class of the java.io package is an abstract superclass that represents an output stream of bytes. OutputStream is an abstract class, it is not useful by itself. However, its subclasses can be used to write data.

1. What are the methods of OutputStream?

Ans: The methods of OutputStream are:

* + - write() - writes the specified byte to the output stream.
    - write(byte[] array) - writes the bytes from the specified array to the output stream.
    - flush() - forces to write all data present in the output stream to the destination.
    - close() - closes the output stream.

1. What is serialization in Java?

Ans: Serialization is the process of converting an object into a stream of bytes to transfer it over a network or to store it in a file or database. In Java, serialization is done by implementing the Serializable interface.

1. What is the Serializable interface in Java?

Ans: The Serializable interface in Java is a marker interface that has no methods. It is used to mark classes that can be serialized, meaning their object instances can be converted into a stream of bytes.

1. What is deserialization in Java?

Ans: Deserialization is the process of converting a stream of bytes back into an object instance. This is done after an object has been serialized.

1. How is serialization achieved in Java?

Ans: Serialization is achieved in Java by implementing the Serializable interface. When an object is serialized, its state is converted into a stream of bytes, which can then be transferred over a network or stored in a file or database.

1. How is deserialization achieved in Java?

Ans: Deserialization is achieved in Java by reading a stream of bytes and using them to recreate the original object instance. This is done by calling the readObject() method of an ObjectInputStream instance.

1. How can you avoid certain member variables of class from getting serialized?

Ans: Mark member variables as static or transient and those member variables will no more be a part of Serialization.

1. What classes are available in the Java IO File Classes API?

Ans: The following classes are available in the Java IO API and are important to work with files in Java.

1. File
2. RandomAccessFile
3. FileInputStream
4. FileReader
5. FileOutputStream
6. FileWriter
7. What is Difference between Externalizable and Serialization interface?

Ans:The Difference between Externalizable and Serialization interface are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Key** | **Serialization** | **Externalization** |
| 1 | Interface | Serialization is a marker interface | Externalization contains two methods readExternal and writeExternal. |
| 2 | Implementation logic | The class which is implementing this interface gives the responsibility to JVM for serializing or persist java object.  JVM use readObject and writeObject for serialization | Externalization provides implementation logic control to the application by overriding readExternal and writeExternal methods. |
| 3 | Way to ignore variables | In serialization, JVM ignores transient variable during serialization and deserialization of java object | Programmer can write their own logic to ignore some of the variables during externalization of java object |
| 4 | Performance | In serializable interface uses reflection which causes relatively slow performance. | Externalizable gives full control over the implementation approach. |
| 5 | Object serialization with inheritance | 1. If the superclass is not serializable then the subclass still can be serialized. 2. If a subclass is not serialized but superclass is automatically serializable | We can apply this to externalizable as well. |