**OOPS**

**CLASS:**

**A Class is a blueprint for creating an object.**

**OBJECT:**

**An object is an instantiation of a class. Memory is allocated only after object instantiation.**

**PROPERTIES OF OOPS:**

1. **ABSTRACTION: It is a feature of OOPs that allows to show only essential details and hiding rest of the details from user.**
2. **ENCAPSULATION: Wrapping various variables and functions together.**
3. **INHERITANCE: It mainly helps in reusability of code.**
4. **POLYMORPHISM: One entity many forms.**

**ACCESS MODIFIERS:**

**As the name suggests access modifiers in Java helps to restrict the scope of a class, constructor, variable, method, or data member. There are three types of access modifiers available in java:**

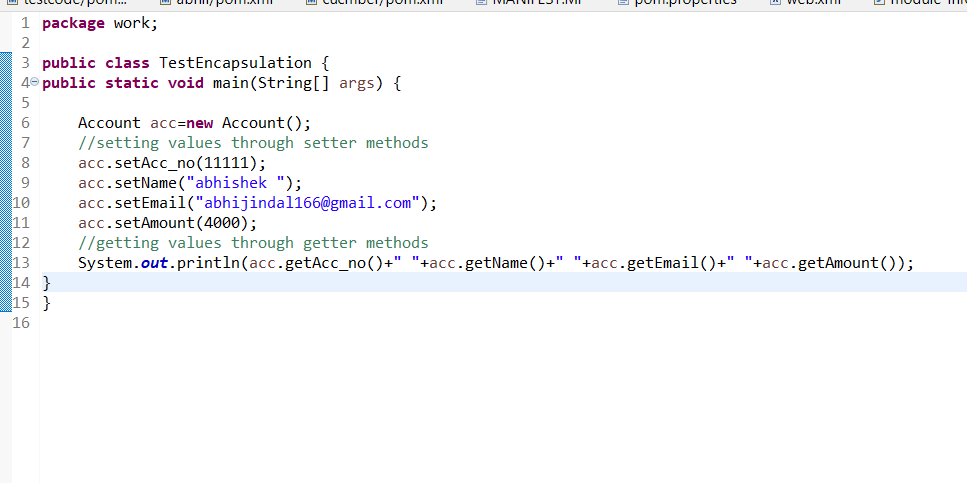
1. **Private**
2. **Protected**
3. **Public**

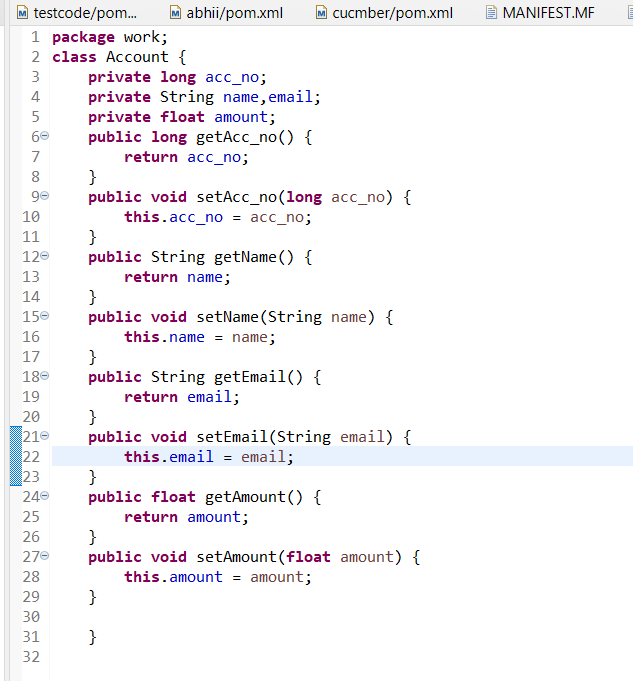
**Getter: Return the value.**

**Setter: Sets the value.**

**Encapsulation**

**It is wrapping up the data into single unit. It acts as a shield that prevent data to be accessed outside the shield. In encapsulation**  **the variables or data of a class is hidden from any other class and can be accessed only through any member function of its own class in which it is declared.**

****

****

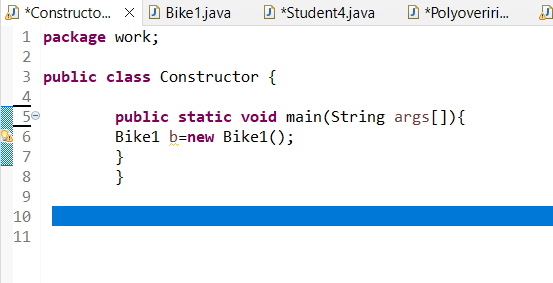
**Constructors**

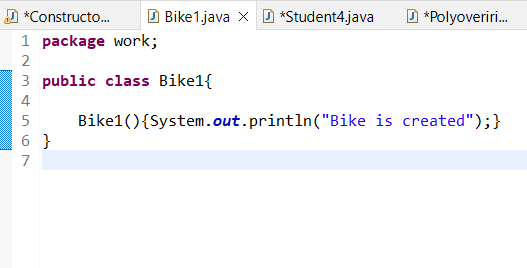
**A member function used to initialize an object while creating it.**

**In order to write our own constructor we define a method with same name as class name.**

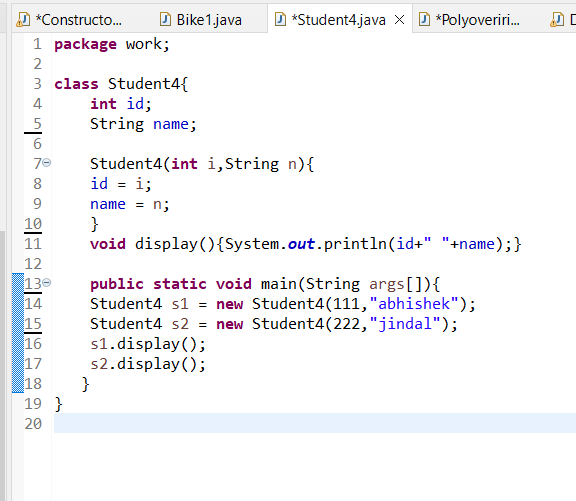
**Constructor can be both parameterized and non-parameterized.**

**Non-Parameterized**

****

****

**Parameterized**

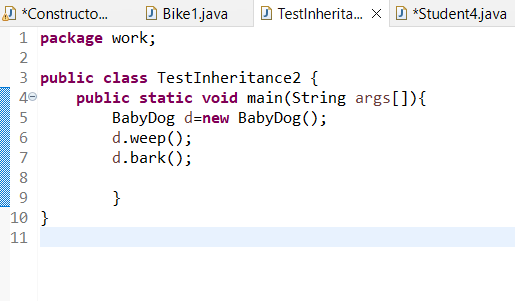
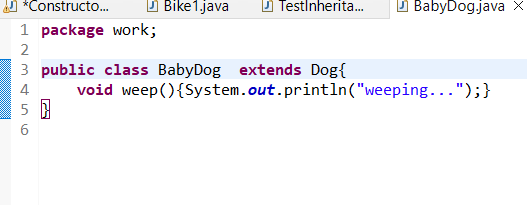
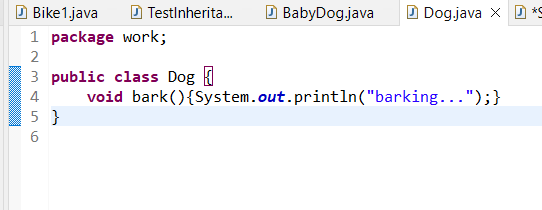
****

**INHERITANCE**

**It is basically inheriting all properties of parent class.**

**Eg. Class B inherits Class A then class B will have all properties of Class A.**

**It is mainly used to create new classes from existing classes.**

**** **** 

**CONSTRUCTOR IN INHERITANCE**

**When derived class is extended from base class the constructor of base class is executed first followed by derived class.**

# ‘this’ reference in Java

**‘this’ is a reference variable that refers to current object.**

**class Test**

**{**

**int a;**

**int b;**

**Test(int a, int b)**

**{**

**this.a = a;**

**this.b = b;**

**}**

**}**

**‘Super’ Keyword**

**It is used to refer immediate parent.**

**If we want to call parameterized constructor from parent class we use super keyword.**

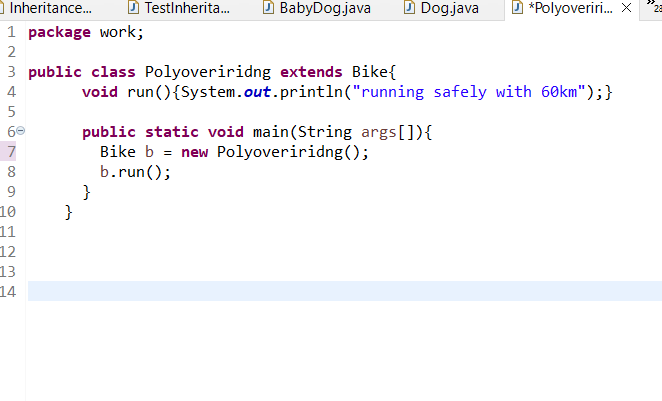
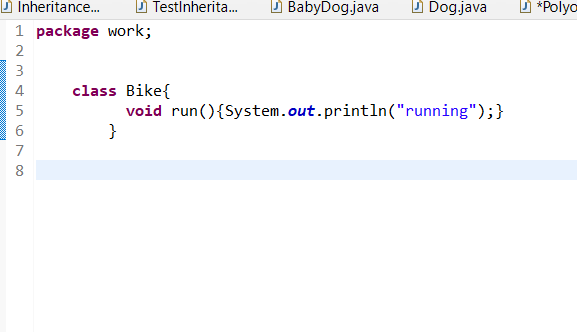
**POLYMORPHISM**

**In Java polymorphism is mainly divided into two types:**

* **Compile-time Polymorphism(Method overloading)**
* **Runtime Polymorphism(Method overriding)**

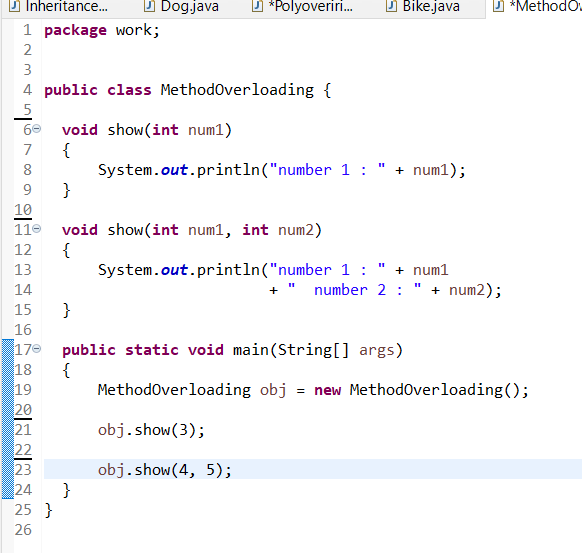
**METHOD OVERRIDING**

**If subclass has the same methods as declared in parent class.**

**** ****

**METHOD OVERLOADING**

**A class has multiple methods with same name but different parameters.**

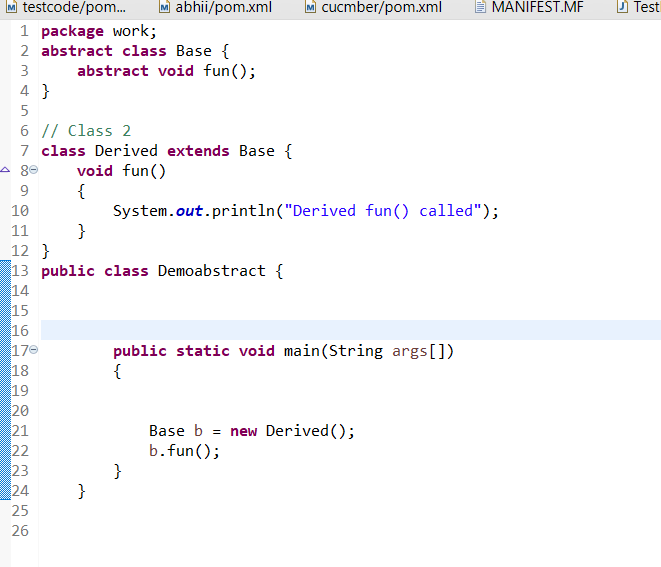
****

**ABSTRACT CLASS**

**ABSTRACT METHOD: A method that is declared without implementation.**

**If a class includes abstract method then it is known as abstract class.**

**It is possible to create reference of abstract class but not an object.**

****

**INTERFACES IN JAVA**

**It is group of related methods with empty bodies. In interfaces all the methods need to be defined by the class implementing it.**

**More than one interface can be implemented by a class.**

**JAVA COLLECTIONS**

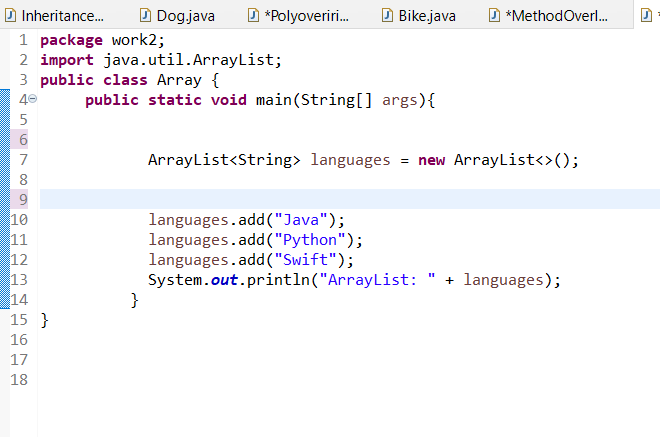
**Collection in java is basically framework that is used to store and change group of objects.**

**LIST INTERFACE:**

**This interface is dedicated to the data of the list type in which we can store all the ordered collection of the objects**.

1. **ARRAY LIST**

**It is a dynamic array in java that is it has no fixed size limit. Java ArrayList allows us to randomly access the list because it works on index basis.**

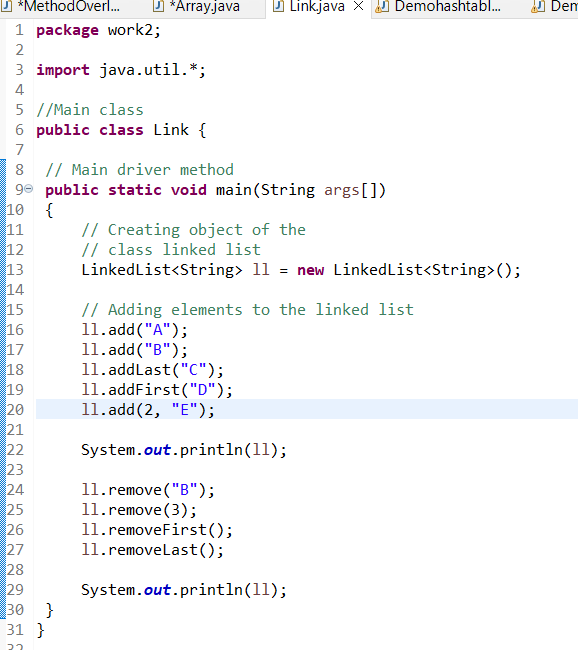
****

1. **LINKED LIST**

**In this the elements are not stored in contiguous locations and every element is a separate object with a data part and address part.**

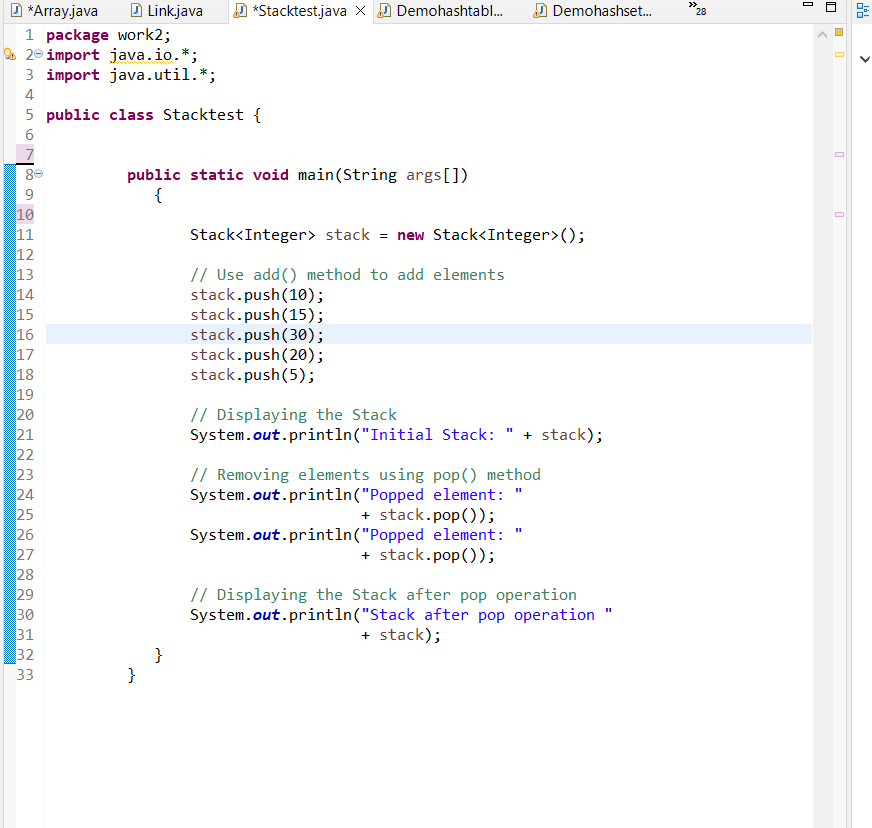
**As it doesn’t work on index basis so we cannot randomly access it.**

**It is faster then array list as no shifting is required.**

****

1. **STACK**

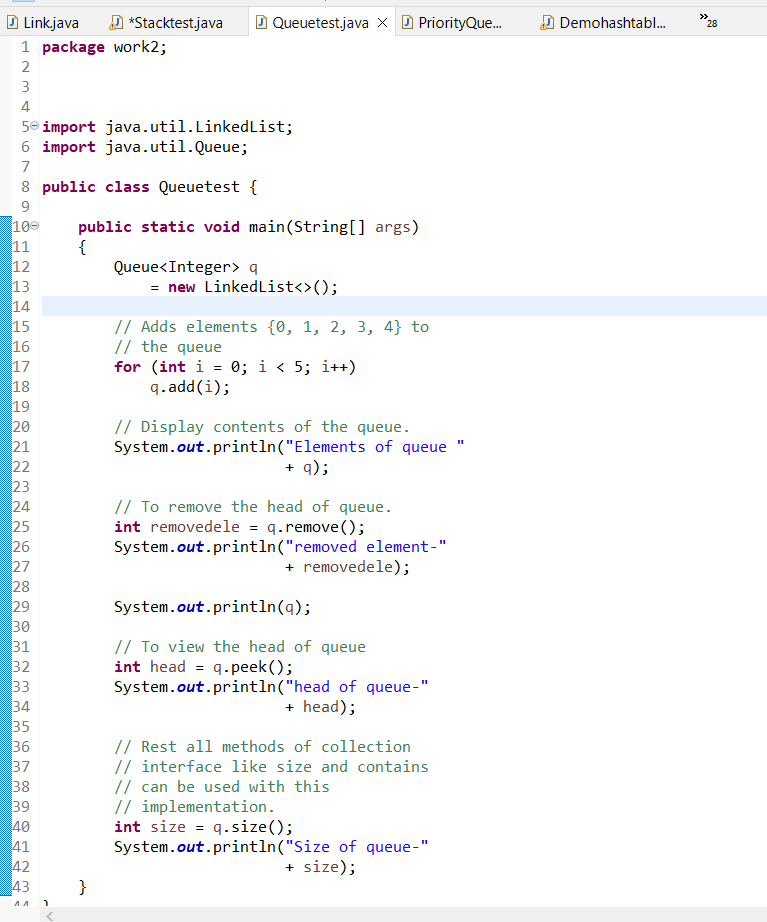
**The class is based on the basic principle of *last-in-first-out*. It contain methods push, peek, pop, size etc.**

****

**QUEUE INTERFACE**

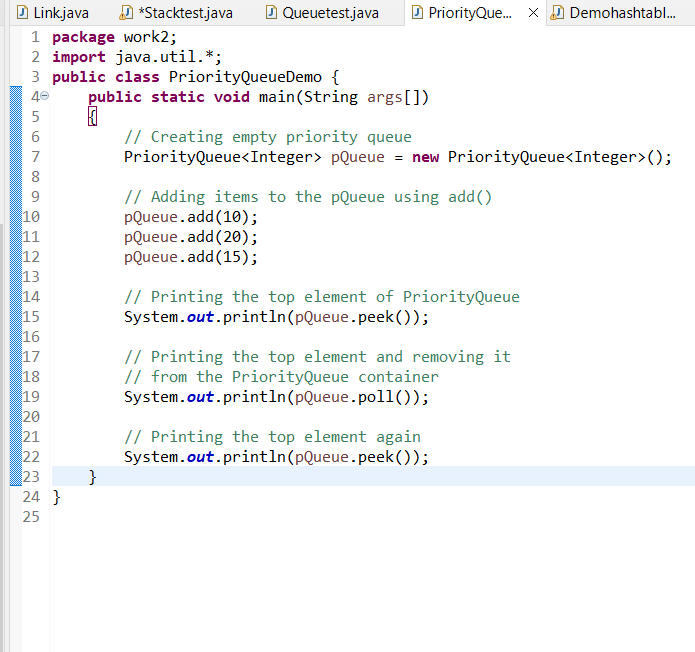
**A queue interface maintains the FIFO(First In First Out) order.**

**It contain methods like add, remove, peek, size.**

****

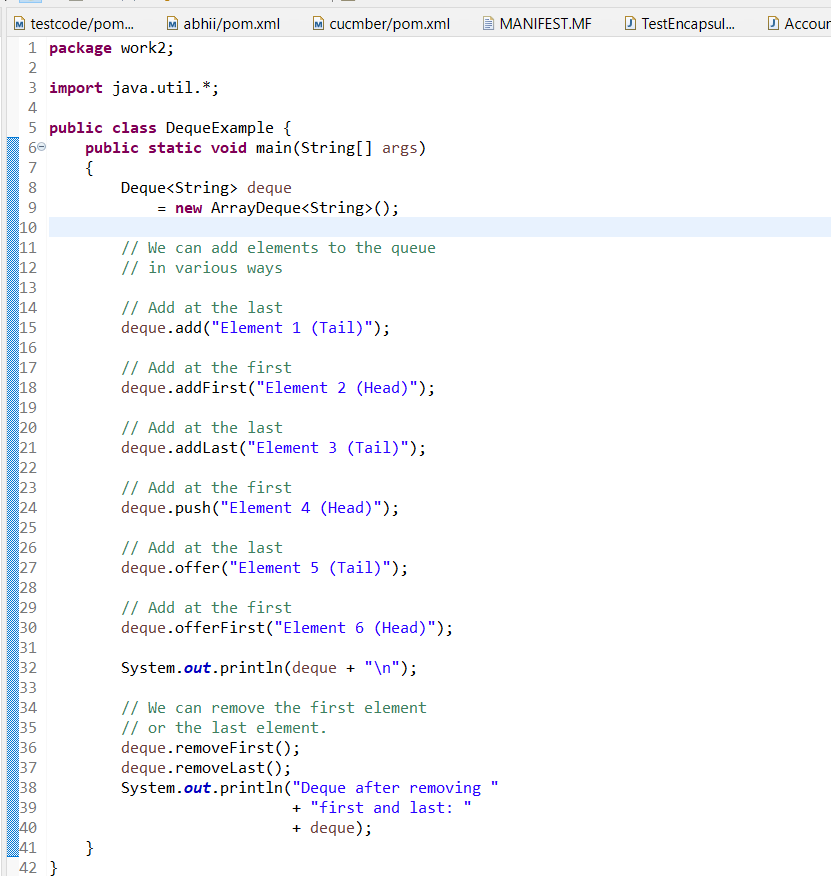
**PRIORITY QUEUE:**

**A Priority Queue is used when the objects are supposed to be processed based on the priority. The Priority Queue is based on the priority heap.**

****

**DEQUEUE INTERFACE**

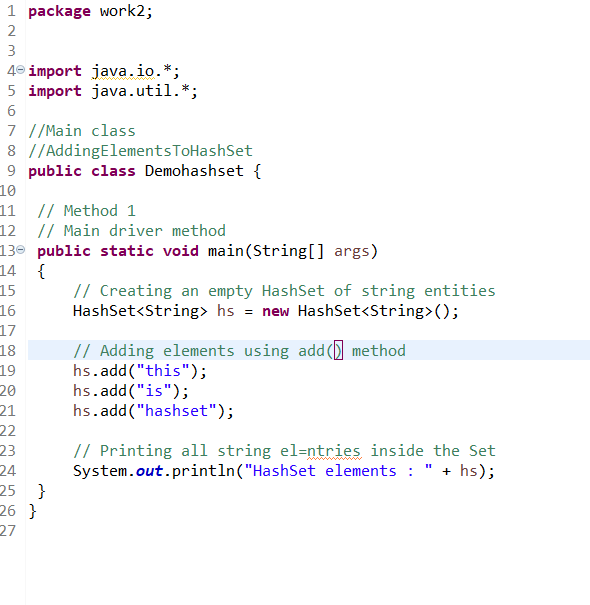
**Double-ended queue, is a data structure where we can add and remove the elements from both ends of the queue.**

****

**HASH SET**

**The objects that we insert into the Hash Set do not guarantee to be inserted in the same order. This collection is used when we wish to avoid the duplication of the objects.**

**It has basic operations like add, remove, contain and size.**

****

**HASHMAP**

**It stores the data in (Key, Value) pairs. To access a value in a Hash Map, we must know its key. If we try to insert a duplicate key, it will replace the element to the corresponding key.**

