Object Class

13 February 2023 08:02

Object Class :

- Object Class is present in java.lang package
- Object Class is the super-most class in java. in other words if we create any object, Object class members are loaded first
- There are total 11 methods in Object class

Return type	Method Name
Class	getClass()
boolean	equals(Object o)
int	hashCode()
Object	clone()
String	toString()
void	finalize

Used in multi-threading

Return Type	Method Name
void	wait()
void	wait(long miliSeconds)
void	wait(long ms, int x)
void	notify()
void	notifyAll()

Equals Method

13 February 2023 08:16

Equals Method

- Object Class's Equal method compares two objects on the basis of address
- if we have to compare two objects on the basis of states override equals method

Console

```
<terminated> C [Java Application] C:\Progra
false
true
```

Main.java Car.java

```java
package abstraction;

public class Car {
    int price;
    String brand;

    public Car(int price , String model) {
        this.price = price;
        this.brand = model;
    }

    public boolean equals(Object obj) {
        Car c = (Car) obj;
        return(this.price == c.price && this.brand.equals(c.brand))
    }

}

class C{
    public static void main(String[] args) {
        Car c1 = new Car(6000,"ford");
        Car c2 = new Car(6000,"ford");

        System.out.println(c1 == c2);
        System.out.println(c1.equals(c2));
    }
}
```

- here the first print statement compares the address of c1 and c2
- here the second print statement compare the states of c1 and c2

Hash Code

13 February 2023 08:41

hashCode :

- hashCode() is responsible for allocating an address to an object
- in other words whenever new keyword is used hashCode method is called internally
- hashCode generates unique integer numbers to every object

Console

```
<terminated> C [Java Application] C:\Progra
true
true
3335948
3335948
94783644
94783644
```

Main.java | Car.java

```java
package abstraction;

import java.util.Objects;

public class Car {
    int price;
    String brand;

    public Car(int price , String model) {
        this.price = price;
        this.brand = model;
    }

    public boolean equals(Object obj) {
        Car c = (Car) obj;
        return(this.price == c.price && this.brand.equals(c.brand));
    }

    public int hashCode() {
        return Objects.hash(price, brand);
    }

}

class C{
    public static void main(String[] args) {
        Car c1 = new Car(6000,"ford");
        Car c2 = new Car(6000,"ford");
        Car c3 = new Car(5000,"chevy");
        Car c4 = new Car(5000,"chevy");

        System.out.println(c1.equals(c2));
        System.out.println(c3.equals(c4));
        System.out.println(c1.hashCode());
        System.out.println(c2.hashCode());
        System.out.println(c3.hashCode());
        System.out.println(c4.hashCode());
    }
}
```

- here we have overridden equals and hashCode method that they don't allow duplicate object creation with other memory location, to follow the equality contract , whenever a duplicate object is created it will be pointed at the same memory location

Equality Contract

13 February 2023 09:10

Equality Contract

- if two objects are declared same or identical by equals method then the hash codes of the two objects should also be same, hence whenever we override equals() method we also have to override hashCode method

  Assignment - create 4 student objects with same states. If equals method says true hashCode() method should not create another object

toString method

14 February 2023 07:17

toString() method :

- toString() converts hash-code of an object to hexa-decimal format
- this method is responsible for printing hexadecimal address rather than hash-code

Car.java × Cow.java

```java
    package abstraction;

import java.util.Objects;

public class Car {
    int price;
    String brand;

    public Car(int price , String model) {
        this.price = price;
        this.brand = model;
    }

    public boolean equals(Object obj) {
        Car c = (Car) obj;
        return(this.price == c.price && this.brand.equals(c.brand));
    }

    public int hashCode() {
        return Objects.hash(price, brand);
    }

//  public String toString() {
//      return price + " " + brand;
//  }

}

class C{
    public static void main(String[] args) {
        Car c1 = new Car(6000,"ford");
        Car c2 = new Car(6000,"ford");

        System.out.println(c1);
        System.out.println(c2);
    }
}
```

Console ×

```
<terminated> C [Java Application] C:\F
abstraction.Car@32e70c
abstraction.Car@32e70c
```

Car.java × Cow.java

```java
    package abstraction;

import java.util.Objects;

public class Car {
    int price;
    String brand;

    public Car(int price , String model) {
        this.price = price;
        this.brand = model;
    }

    public boolean equals(Object obj) {
        Car c = (Car) obj;
        return(this.price == c.price && this.brand.equals(c.brand));
    }

    public int hashCode() {
        return Objects.hash(price, brand);
    }

    public String toString() {
        return price + " " + brand;
    }

}

class C{
    public static void main(String[] args) {
        Car c1 = new Car(6000,"ford");
        Car c2 = new Car(6000,"ford");

        System.out.println(c1);
        System.out.println(c2);
    }
}
```

Console ×

```
<terminated> C [Java Ap
6000 ford
6000 ford
```

clone method

14 February 2023 07:39

clone method :

- clone method is used to get an exact copy of an object

rules :

- if we want to use clone method a class should implement cloneable interface otherwise we'll get clone not supported exception error.
- since clone method is a protected method it has to be overridden to clone an object

Car.java × Cow.java

```java
    package abstraction;

import java.util.Objects;

public class Car implements Cloneable {
    int price;
    String brand;

    public Car(int price , String model) {
        this.price = price;
        this.brand = model;
    }

    public boolean equals(Object obj) {
        Car c = (Car) obj;
        return(this.price == c.price && this.brand.equals(c.brand));
    }

    public int hashCode() {
        return Objects.hash(price, brand);
    }

    public String toString() {
        return price + " " + brand;
    }
    protected Object clone() throws CloneNotSupportedException{
        return super.clone();
    }

}

class C{
    public static void main(String[] args)throws CloneNotSupportedExce
        Car c1 = new Car(1200000,"Mahindra");
        Car c2 = (Car) c1.clone();

        System.out.println(c2.price);
        System.out.println(c2.brand);

    }
}
```

Console ×

```
<terminated> C [Java App
1200000
Mahindra
```

Wrapper Class

14 February 2023 08:22

Wrapper Classes :

- Wrapper class is used to convert primitive to non-primitive and vice versa
- All the wrapper classes are present in java.lang package
- All the wrapper classes are final classes
- All the wrapper classes are implementing comparable interface.

Primitive	Corresponding Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Autoboxing

14 February 2023 08:42

Autoboxing :

- automatic conversion from primitive to non-primitive

Auto-unboxing :

- automatic conversion from non-primitive to primitive

Parsing methods

14 February 2023 08:47

- Parsing methods are used to convert string representation of a number to actual number
- we have to pass a valid string with only numbers to parsing methods, else these methods throw number format exception

Console

<terminated> B [Java Appli

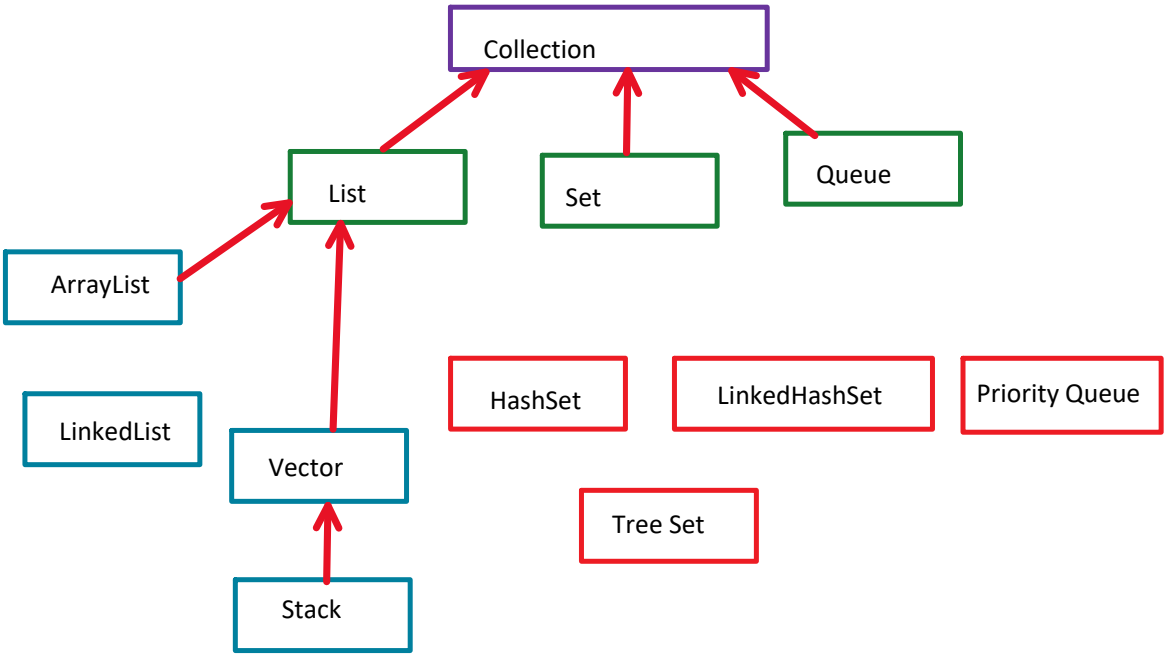
```
123
1234.57
```

B.java

```java
package abstraction;

public class B {
    public static void main(String[] args) {
        int val = Integer.parseInt("123");
        System.out.println(val);
        float f = Float.parseFloat("1234.57");
        System.out.println(f);

    }
}
```

Collections Framework

14 February 2023 08:47



Collection Framework

- Collection framework is a readymade utility in java from jdk 1.2
- Collection is an interface present in java.util package

List

- List is a sub-interface of collection framework
- List is index based.
- List maintains insertion order
- List can store duplicate elements
- List can store multiple null values

note :

- any collection is not fixed in size hence we overcome the disadvantage of an array
- collections can be homogeneous as well as heterogeneous
- collections cannot store primitive data. It can only store objects

some methods in collection framework

- all these methods are public and abstract

boolean	**add(Object o)**
boolean	**addAll(Collection c)**
boolean	**remove(Object o)**
int	**size()**
boolean	**isEmpty()**
void	**clear()**
boolean	**contains(Object o)**
boolean	**contains(Collection c)**
boolean	**removeAll(Collection c)**
Object[]	**toArray()**
Integer	**iterator()**

Console

<terminated> A (1) [Jav

```
2
false
0
true
```

B.java A.java

```java
package collectionFrameworks;

import java.util.ArrayList;
import java.util.Collection;

public class A {

    public static void main(String[] args) {

        Collection al = new ArrayList();
        al.add(10);
        al.add(25.789);
        System.out.println(al.size());
        System.out.println(al.isEmpty());
        al.clear();
        System.out.println(al.size());
        System.out.println(al.isEmpty());

    }

}
```

al.add() : adds an element to the array list al
al.size() : returns the size of the arrayList
al.isEmpty : tells if the arrayList is empty or not
al.clear() : removes all the elements from the arrayList

Methods in List Interface

15 February 2023 08:11

methods in List interface

boolean	add(int index,Object o)
boolean	addAll(int index,Collection c)
<E>	remove(int index)
<E>`	get(int index)
int	indexOf(Object o)
int	lastIndexOf(Object o)
ListIterator	listIterator()
ListIteration	listIterator(int index)
<E>	set(int index, Object o)

ArrayList

15 February 2023 08:19

ArrayList :

- It follows global array data structure

points to remember :

- the initial capacity of an arrayList is 10
- the incremental capacity is $\frac{Current\ Capacity \times 3}{2} + 1$

```
for(int i = 1; i <= 10 ; i++) {
    al.add(i*10);
}
al.add(120);
```

10	20	30	40	50	60	70	80	90	100	123
0	1	2	3	4	5	6	7	8	9	
10										

- here the default capacity of the array is 10 at first
- after the loop is done running the element 120 is to be added at 10th index so the size needs to be incremented
- the size of the array list increments by 10x3/2 + 1 that is 16 and then 16x3/2 + 2 that is 25

For-Each Loop

16 February 2023 07:20

syntax :

```
for( Type of element in Collection  VarName : collection reference ){
        //statements;
}
```

Console

```
<terminated> A (1) [Java Application] C:\Program
2 3 4 5 6 0
5 10 15 20 25 30 35 40 45 50
```

B.java A.java

```java
package collectionFrameworks;

import java.awt.List;

public class A {

    public static void main(String[] args) {

        int a[] = { 2, 3, 4,5, 6, 0 };
        for(int i : a ) {
            System.out.print(i + " ");
        }

        ArrayList al = new ArrayList();
        for(int i = 1 ; i <= 10 ; i++) {
            al.add(i*5);
        }
        System.out.println();
        for(Object o : al) {
            System.out.print(o + " ");
        }


    }

}
```

for	for-each
for loop was introduced in jdk 1.0	for-each was introduced in jdk 1.5
for loop can be used for any condition based scenarios	for-each loop can only be used with a collection
partial iterations are possible	partial iteration is not possible
using for loop we can traverse any collection in forward or backward direction	using for-each we can traverse only in forward direction

Generics

16 February 2023 07:41

Generics

- Generics are classes that are used to define the type of the collection
- Using Generics we can make a collection homogenous
- If we don't write any Generics the default type would be Object

Console

```
<terminated> A (1) [Java Application] C:\Program
virat sachin dhoni
12 34 55
```

B.java A.java

```java
package collectionFrameworks;

import java.awt.List;

public class A {

    public static void main(String[] args) {

    ArrayList<String> iplteam = new ArrayList<>();
    iplteam.add("virat");
    iplteam.add("sachin");
    iplteam.add("dhoni");
    for(String s : iplteam) {
        System.out.print(s + " ");
    }
    System.out.println();
    ArrayList<Integer> a = new ArrayList<>();
    a.add(12);
    a.add(34);
    a.add(55);
    for(int i : a ) {
        System.out.print(i + " ");
    }

    }

}
```

Linked List

16 February 2023 08:03

linked list:

- Linked List follows doubly linked List data structure
- There is no initial capacity and no incremental capacity

```
LinkedList ll = new LinkedList();
        ll.add(10);
        ll.add(20);
        ll.add(30);
        ll.add(40);
        ll.add(1,50);
```

we usually don't prefer using arrayList for insertion and deletion operations because these kind of operations will lead to shifting of elements hence arrayList is a bad choice for insertion and deletion operations but arrayLists is highly preferred for searching and sorting operations

- linkedList is preferred for insertion and deletion operations and not for searching and sorting operations.

Constructors in array list

16 February 2023 08:33

1. ArrayList();
2. ArrayList(int initial Capacity)
3. ArrayList(Collection c)

1. LinkedList()
2. LinkedList(Collection c)

A.java

```java
package collectionFrameworks;

import java.awt.List;

public class A {

    public static void main(String[] args) {

    ArrayList<String> iplteam = new ArrayList<>();
    iplteam.add("virat");
    iplteam.add("sachin");
    iplteam.add("dhoni");
    for(String s : iplteam) {
        System.out.print(s + " ");
    }
    System.out.println();
    ArrayList<Integer> a = new ArrayList<>();
    a.add(12);
    a.add(34);
    a.add(55);
    for(int i : a ) {
        System.out.print(i + " ");
    }

    }

}
```

Console *B.java

```
<terminated> A (1) [Java Application] C
virat sachin dhoni
12 34 55
```

Assignment : Create a list to store student objects and print every detail of all the objects using toString method.

Singly Linked List implementation

17 February 2023 07:12

```java
package collectionFrameworks;


public class C {
    public static void main(String[] args) {
        LinkedList ll = new LinkedList();
        for(int i = 1 ; i <= 10 ; i++) {
            ll.add(i*10);
        }
        System.out.println(ll.size());
        ll.display();
    }
}

class LinkedList{
    private int size;
    private Node head,temp;

    public int size(){
        return size;
    }

    public void add(int data) {
        Node n = new Node();
        n.val = data;
        if(head == null) {
            head = n ;
            temp = n ;
        }else {
            temp.next = n;
            temp = n;
        }
        size++;
    }

    public void display() {
        Node n = head;
        while (n != null) {
            System.out.print(n.val + " ");
            n = n.next;
        }
        System.out.println();
    }
}

class Node {
    int val;
    Node next;
}
```

Stack

17 February 2023 08:01

Stack

- Stack follows first in last out (FILO) data structures

important methods of Stack

1. push(Object o) // adds an object on top of the stack
2. peek() // returns the element from top of the stack
3. pop() // returns the element form top of the stack and removes it
4. isEmpty()

Q. WAP to check whether brackets are balanced or not,,

Vector

19 February 2023 18:20

Vector :

- Vector is similar to array but it is not muti-threaded
- The initial capacity is 10 , incremental capacity is **current capacity x 2**

note : Stack, Vector, Hashtable are called as legacy classes of java

ArrayList	**Vectors**	**LinkedList**
initial capacity : 10	initial capacity : 10	n/a
incremental Capacity : $\frac{Current\ Capacity \times 3}{2} + 1$	$Current\ Capacity\ \times 2$	n/a
Growable Array	Growable Array	Doubly LinkedList
3 constructors	4 constructors	2 constructors
Not Thread safe	Is Thread Safe	Not Thread Safe

Set

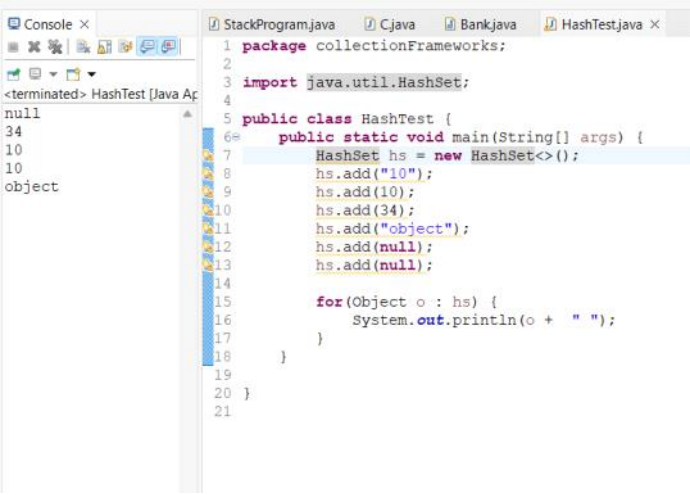
19 February 2023 18:47

Set :

- Set is a chid interface of collection which allows not duplicate values

Characteristics of Set

- It cannot store duplicate elements
- Insertion order is not maintained
- Set is not index based
- At most any set can store maximum of 1 null value



Hash Set

19 February 2023 19:06

Hash Set

- Hash Set is implemented using hash-table data-structure
- Hash table algorithm uses hash-code method internally
- The initial capacity of hash-set is 16
- The load factor is 75%

Console

<terminated> HashSetTest [Java

```
10 4 14 8 2
50 20 40 10 30
```

HashSetTest.java

```java
package collectionFrameworks;
import java.util.HashSet;

public class HashSetTest {

    public static void main(String[] args) {
        HashSet hs = new HashSet<>();
        int a[] = new int[5];
        for (int i = 1 ; i <= 5 ; i++) {
            Object o = i*10;
            a[i-1] = o.hashCode()%16;
            hs.add(o);
        }
        for(int i : a) {
            System.out.print(i + " ");
        }
        System.out.println();
        for(Object o : hs) {
            System.out.print(o + " ");
        }

    }

}
```

HashSet constructors :

- HashSet()
- HashSet(java.util.Collection<? extends E>);
- HashSet(int, float)
- HashSet(int)

Linked Hashset

21 February 2023 07:21

Linked Hashset:

- linked hashset is implemented using hashtable and doubly linked-list data structure.
- linked hashset maintains insertion order
- Since there are two data structures are implemented the memory consumption is quite high

TreeSet

21 February 2023 07:38

TreeSet :

- TreeSet implements balanced tree data structure
- TreeSet maintains natural sorting order(ascending order)
- you can only add comparable type elements to TreeSet
- TreeSet can only store homogenous objects
- If we try to add a heterogenous element we will get class cast exception
- TreeSet cannot store even a single null(if it' s done we'll get null pointer exception)

Console

<terminated> TreeSetTtest [Ja

```
-99 34 123
```

TreeSetTtest.java

```java
package collectionFrameworks;

import java.util.TreeSet;

public class TreeSetTtest {
    public static void main(String[] args) {
        TreeSet ts = new TreeSet<>();
        ts.add(123);
        ts.add(34);
        ts.add(-99);
        for(Object o : ts) {
            System.out.print(o + " ");
        }
    }

}
```

Console

<terminated> TreeSetTtest [Java Applicat

```
abc abd ac xy xyz
```

*TreeSetTtest.java

```java
package collectionFrameworks;

import java.util.TreeSet;

public class TreeSetTtest {
    public static void main(String[] args) {
        TreeSet ts = new TreeSet<>();
        ts.add("ac");
        ts.add("abc");
        ts.add("abd");
        ts.add("xy");
        ts.add("xyz");
        //strings will be compared lexicographically
        for(Object o : ts) {
            System.out.print(o + " ");
        }
    }

}
```

Queue

21 February 2023 08:07

- Queue implements first in , first out data-structure

Console

<terminated> QueueTest [Java Applicatio

```
10
10
15
false
true
```

TreeSetTtest.java | QueueTest.java

```java
package collectionFrameworks;

import java.util.PriorityQueue;
import java.util.Queue;

public class QueueTest {
    public static void main(String[] args) {
        Queue q = new PriorityQueue<>();
        q.add(10);
        q.add(15);
        q.add(20);

        Object p = q.peek();
        System.out.println(p);
        p = q.poll();
        System.out.println(p);
        p = q.peek();
        System.out.println(p);
        System.out.println(q.isEmpty());
        q.clear();
        System.out.println(q.isEmpty());
    }

}
```

DataStructures :

21 February 2023 08:34

DataStructures :

1. Linear
    a. Arrays
    b. LinkedList
    c. Queue
    d. Stack
2. Non-Linear
    a. Graph
    b. Tree

Map

21 February 2023 08:37

Map

1. HashMap
2. LinkedHashMap
3. Tree Map
4. HashTable

Methods

22 February 2023 07:20

V	put(K key , V value);
int	size();
boolean	isEmpty();
boolean	containsKey(Object key);
boolean	containsValue(Object value);
V	get(Object key);
V	remove(Object key);
void	Clear();
void	putAll(Map<K,V>m);
Set<k>	keySet();
Collection<V>	values();
Set<Map.Entry<K,V>	entrySet();

```
interface Map{
//methods
interface Entry{
k getKey();
v getValue();
}
}
```

nested interface

Entry :

- Entry is a key and value pair
- Map is group of entries
- key must be unique
- values can be duplicated in any map,

k1-v1	k2-v2	k3-v3	k4-v4	k5-v5	k6-v6

a group of 6 entries, a map

HashMapTest.java

```java
package maps;

import java.util.HashMap;
import java.util.Map;

public class HashMapTest {

    public static void main(String[] args) {
        Map m = new HashMap<>();
        m.put(1,345);
        m.put("abc",345);
        m.put("xyz", 345);
        m.put("abc", 345);
        m.put(2, 345);
        m.put(1, 345);
        System.out.println(m);

        Map<Integer,String> m1 = new HashMap<>();
        m1.put(1, "abc");
        m1.put(2, "abc");
        m1.put(2,"xyz");
        m1.put(3, "def");
        m1.put(3, "abc");
        m1.put(3, "hello");
        System.out.println(m1);

    }

}
```

Console | Theatre.java | Ticket.java

```
<terminated> HashMapTest [Java Application] C:\Progra
{1=345, 2=345, abc=345, xyz=345}
{1=abc, 2=xyz, 3=hello}
```

HashMapTest.java

```java
package maps;

import java.util.Collection;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class HashMapTest {

    public static void main(String[] args) {
        Map<String, Integer> rcbteam = new HashMap<>();
        rcbteam.put("viart", 18);
        rcbteam.put("siraj", 34);
        rcbteam.put("padaker", 3);
        rcbteam.put("chahal", 4);

        Map<String,Integer> miteam = new HashMap<>();
        miteam.put("sky", 63);
        miteam.put("rohit", 45);
        miteam.put("bumrah", 27);
        miteam.put("chahal", 4);

        Map<String,Integer> indiateam = new HashMap<>();
        indiateam.putAll(rcbteam);
        indiateam.putAll(miteam);
        System.out.println(rcbteam);
        System.out.println(miteam);
        System.out.println(indiateam);
        System.out.println(indiateam.containsKey("virat"));
        System.out.println(indiateam.containsValue(45));
        Integer v = indiateam.get("chahal");
        System.out.println(v);
        v = indiateam.remove("padaker");
        System.out.println(v);
        v = indiateam.remove("xyz");
        System.out.println(v);

        Set<String> keys = indiateam.keySet();
        System.out.println(keys);
        Collection<Integer> values = indiateam.values();
        System.out.println(values);
    }
```

Console | Theatre.java | Ticket.java | C.java

```
<terminated> HashMapTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (22-Feb-2023, 8:48:5
{viart=18, chahal=4, siraj=34, padaker=3}
{sky=63, bumrah=27, chahal=4, rohit=45}
{viart=18, sky=63, chahal=4, bumrah=27, siraj=34, padaker=3, rohit=45}
false
true
4
3
null
[viart, sky, chahal, bumrah, siraj, rohit]
[18, 63, 4, 27, 34, 45]
```

Iterator

23 February 2023 07:13

- Iterator is an interface present in java.util package
- technically iterator is a cursor present at the beginning of a collection
- iterator is having three methods

boolean	hasNext();
<E>	next();
void	remove();

IteratorTest.java

```java
package iterator;

import java.util.HashSet;
import java.util.Iterator;

public class IteratorTest {
    public static void main(String[] args) {
        HashSet<String> hs = new HashSet<>();
        for(int i = 1 ; i <= 6 ; i++) {
            hs.add(i+"abc");
        }
        System.out.println(hs);
        Iterator<String> i = hs.iterator();
        while(i.hasNext()) {
            System.out.println(i.next());
        }
    }

}
```

Console

```
<terminated> IteratorTest [Java Application] C:\Program File
[5abc, 1abc, 6abc, 3abc, 4abc, 2abc]
5abc
1abc
6abc
3abc
4abc
2abc
```

traversing an Array-List in reverse order using listIterator methods

IteratorArrayList.java

```java
package iterator;

import java.util.LinkedList;
import java.util.ListIterator;

public class IteratorArrayList {

    public static void main(String[] args) {
        LinkedList<Integer> ll = new LinkedList<>();
        for(int i = 1 ; i <= 4 ; i++) {
            ll.add(i);
        }
        System.out.println(ll);
        ListIterator<Integer> i = ll.listIterator(ll.size());
        while(i.hasPrevious()) {
            System.out.println(i.previous());
        }
    }

}
```

Console

```
<terminated> IteratorArrayList [Ja
[1, 2, 3, 4]
4
3
2
1
```

Exception Handling

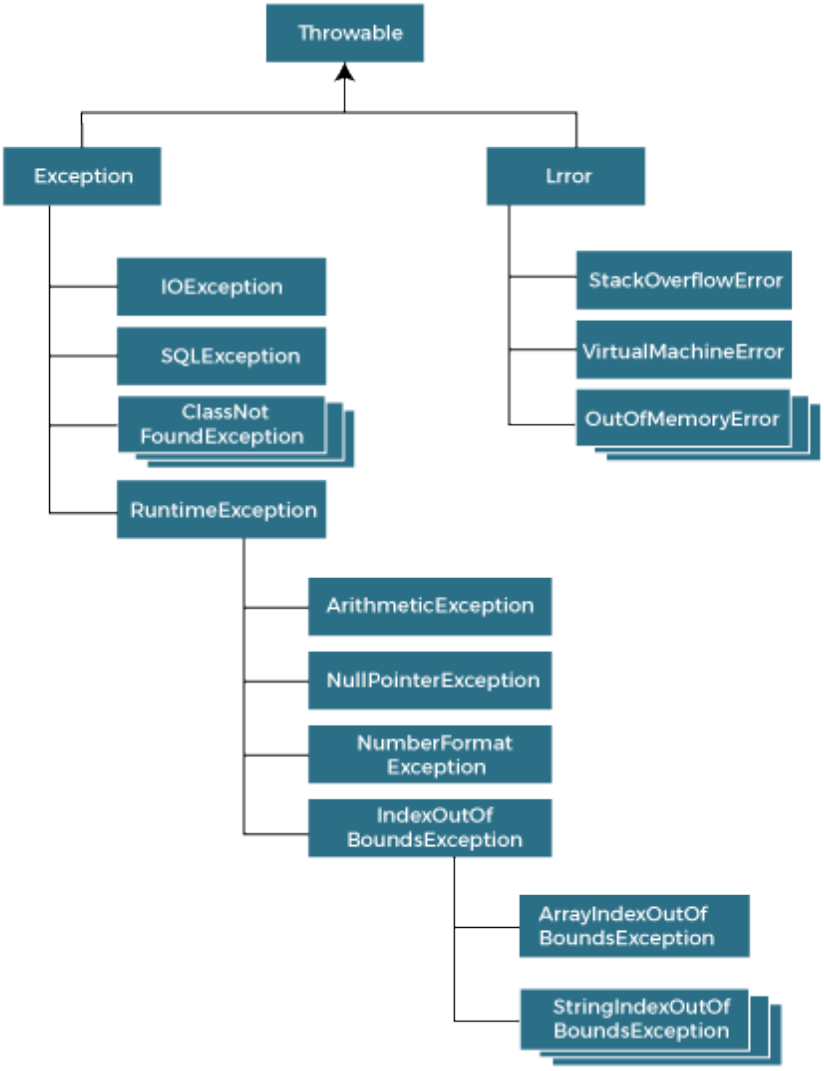
23 February 2023 08:17

Exception

- Exception is a runtime interruption due to which a program stops
- Exception is an unexpected event occurred during runtime

Exception Handling :

- Exception handling deals with resolving an exception



Risky Code

23 February 2023 08:27

The code which might throw an exception is called as risky code

Exception can be handled using :

1. try-catch
2. try-catch-finally

Try Catch Block

- we have to write risky code inside try block
- we cannot write catch block without try block
- we cannot write try block alone
- we can write try block with catch block
- we can write try block with finally block
- or try block with both catch and finally

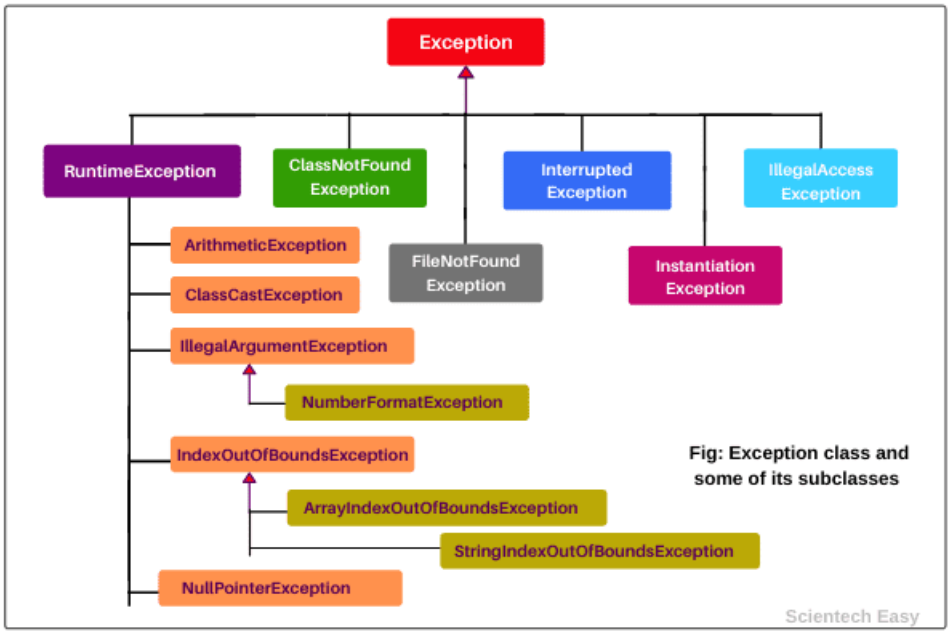


Fig: Exception class and some of its subclasses

Scenario 1

24 February 2023 07:31

Exceptions

- Every exception is an object in java
- only one exception can occur in try block
- the super-most class in exception hierarchy is throwable class

scenario 1 , when there no exception occurs in try block

Console

```
<terminated> TryCatchTest [Java
2
program ended
```

TryCatchTest.java

```java
package exceptionHandling;

public class TryCatchTest {

    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        try {
            int res = a/b;
            System.out.println(res);
        }catch(ArithmeticException e) {
            System.out.println("Catch block is executed");
        }
        System.out.println("program ended");

    }

}
```

Scenario 2 : when there is an exception occurred in try block, the control goes to catch block from the line where exception was occurred.

Console

```
<terminated> TryCatchTest [Java Applicatio
try block
Catch block is executed
program ended
```

TryCatchTest.java

```java
package exceptionHandling;

public class TryCatchTest {

    public static void main(String[] args) {
        int a = 10;
        int b = 0;

        try {
            System.out.println("try block");
            int res = a/b;
            System.out.println(res);
        }catch(ArithmeticException e) {
            System.out.println("Catch block is executed");
        }
        System.out.println("program ended");

    }

}
```

Scenario 3 : When there's an exception occurring in try block but if there is no matching catch block, exception will not be handled

```java
TryCatchTest.java
package exceptionHandling;

public class TryCatchTest {

    public static void main(String[] args) {
        int a = 10;
        int b = 0;

        try {
            System.out.println("try block");
            int res = a/b;
            System.out.println(res);
        }catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Catch block is executed");
        }
        System.out.println("program ended");
```

```
Console
<terminated> TryCatchTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (24-Feb-2023, 8:16
try block
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at exceptionHandling.TryCatchTest.main(TryCatchTest.java:11)
```

Scenario 4 : An exception occurred in child block can be called by a catch block with its parent type

```java
TryCatchTest.java

    public static void main(String[] args) {
        int a = 10;
        int b = 0;

        try {
            System.out.println("try block");
            int res = a/b;
            System.out.println(res);
        }catch(RuntimeException e) {
            System.out.println("Catch block is executed");
        }
        System.out.println("program ended");

    }

}
```

```
Console
<terminated> TryCatchTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (24-Feb-2023, 8
try block
Catch block is executed
program ended
```

Scenario no 5 : we can write multiple catch blocks for single try block

Exception Handling 2

25 February 2023 06:24

if you write multiple catch blocks, catch block with particular exception occurred in try block that will be executed.

Scenario no 6 : when we write multiple catch blocks, the sequence of catch blocks should be from child to parent else compilation error occurs

TryCatchTest.java

```java
package exceptionHandling;

public class TryCatchTest {

    public static void main(String[] args) {
        int a = 10;
        int b = 0;


        try {
            System.out.println("try block");
            int res = a/b;
            System.out.println(res);
        }catch(ArithmeticException e) {
            System.out.println("Airithmetic exception");
        }catch(RuntimeException e) {
            System.out.println("Runtime Exception");
        }

        System.out.println("program ended");

    }

}
```

Console

```
<terminated> TryCatchTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (25-Feb
try block
Airithmetic exception
program ended
```

Finally

25 February 2023 06:39

Finally :

- finally block can be written with try block or try catch block
- usually finally block is used to close costly resources or system resources
- finally block will be executed at any cost irrespective of the exception

TryCatchTest.java

```java
package exceptionHandling;
import java.util.Scanner;

public class TryCatchTest {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter a number");


        try {
            int a = sc.nextInt();
            System.out.println("try block");
            int res = 10/a;
            System.out.println(res);
        }catch(ArithmeticException e) {
            System.out.println("Airithmetic exception");
        }finally{
            System.out.println("Finally Block");
        }

        System.out.println("program ended");

    }
```

Console

```
<terminated> TryCatchTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (25-Feb-
enter a number

0
try block
Airithmetic exception
Finally Block
program ended
```

- when there is no catch block we will get the said exception but the finally block will also be executed

TryCatchTest.java

```java
package exceptionHandling;
import java.util.Scanner;

public class TryCatchTest {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter a number");


        try {
            int a = sc.nextInt();
            System.out.println("try block");
            int res = 10/a;
            System.out.println(res);
        }finally{
            System.out.println("Finally Block");
        }

        System.out.println("program ended");

    }

}
```

Console

```
<terminated> TryCatchTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (25-Feb-2023, 7:07:26 am
enter a number

0
try block
Finally Block
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at exceptionHandling.TryCatchTest.main(TryCatchTest.java:14)
```

Throws

25 February 2023 07:13

Throws :

- It is keyword which is used to indicate an exception
- Throws keyword does not revolve an exception rather it just indicates there is a possibility of an exception that can occur

note : throws keyword can be used with

1. Method declaration
2. Constructor declaration

Checked exception

27 February 2023 06:24

Checked Exception (compile time exception / unhandled exception)

- check exception is that kind of exception where compiler checks whether the exception is handled or not
  in other words, handling a checked exception is mandatory.
- examples :
  - FileNotFoundException
  - ClassNotFoundException
  - IOException

Exception propagation :

- exception traverse from one method to another method is called as exception propagation.

CheckedExceptionTest.java

```java
package exceptionHandling;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

public class CheckedExceptionTest {

    public static void main(String[] args)throws FileNotFoundException{
        m1();
        System.out.println("main end");

    }

    public static void m1()throws FileNotFoundException{
        File f = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\asignment.txt");
        FileInputStream fis = new FileInputStream(f);
        System.out.println("method end");
    }

```

Console

```
<terminated> CheckedExceptionTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (27-Feb-2023, 7:06:15 am – 7:06:15 am) [pid: 100
Exception in thread "main" java.io.FileNotFoundException: C:\Users\samwa\OneDrive\Desktop\asignment
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:158)
        at exceptionHandling.CheckedExceptionTest.m1(CheckedExceptionTest.java:17)
        at exceptionHandling.CheckedExceptionTest.main(CheckedExceptionTest.java:10)
```

CheckedExceptionTest.java

```java
package exceptionHandling;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

public class CheckedExceptionTest {

    public static void main(String[] args)throws FileNotFoundException{
        m1();
        System.out.println("main end");

    }

    public static void m1()throws FileNotFoundException{
        File f = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\assignment.txt");
        FileInputStream fis = new FileInputStream(f);
        System.out.println("method end");
    }

```

Console

```
<terminated> CheckedExceptionTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (27-Feb-2023, 7:06:55 am – 7:06:56 a
method end
main end
```

Unchecked exception(runtime exception / handled exception)

- It is a type of exception where the compiler does not force you to handle it.

Custom exception

27 February 2023 07:20

Custom Exceptions

- Custom exceptions means user defined exceptions
- We can create our own checked type or unchecked type exceptions

common methods used in Custom Exceptions

String	*getMessage()*
void	*printStackTrace()*

InsufficientFundsException.java

```java
package exceptionHandling;

public class InsufficientFundsException extends RuntimeException {

    public String getMessage() {
        return "less amount";
    }

}
```

Console

```
<terminated> A (2) [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (27-Feb-2023, 7:56:04 am – 7:56:09 ar
enter the transfer amount
200
Exception in thread "main" exceptionHandling.InsufficientFundsException: less
        at exceptionHandling.A.transferLayout(A.java:19)
        at exceptionHandling.A.main(A.java:8)
```

A.java

```java
package exceptionHandling;

import java.util.Scanner;

public class A {

    public static void main(String[] args) {
        transferLayout();
        System.out.println("program done");

    }

    public static void transferLayout() throws InsufficientFundsException {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter the transfer amount");
        double available = 195.45;
        int amount = sc.nextInt();
        if (amount > available) {
            InsufficientFundsException a = new InsufficientFundsException();
            throw a ;
        }else {
            System.out.println("amount transfer");
        }
    }

}
```

File handling

27 February 2023 07:31

File handling

- File handling deals with reading data from a file and writing data into a file

Console

```
<terminated> A (3) [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (27-Feb-2023, 8:26:30 am – 8:26:30 am) [pid: 982
file is present in the system
end
```

A.java

```java
package fileHandling;

import java.io.File;
import java.io.IOException;

public class A {
    public static void main(String[] args)throws IOException{
        File f = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\assignment.txt");
        if(f.exists()) {
            System.out.println("file is present in the system");
        }else {
            f.createNewFile();
        }
        System.out.println("end");

    }

}
```

if file is not present createNewFile() method creates a new file of the same name

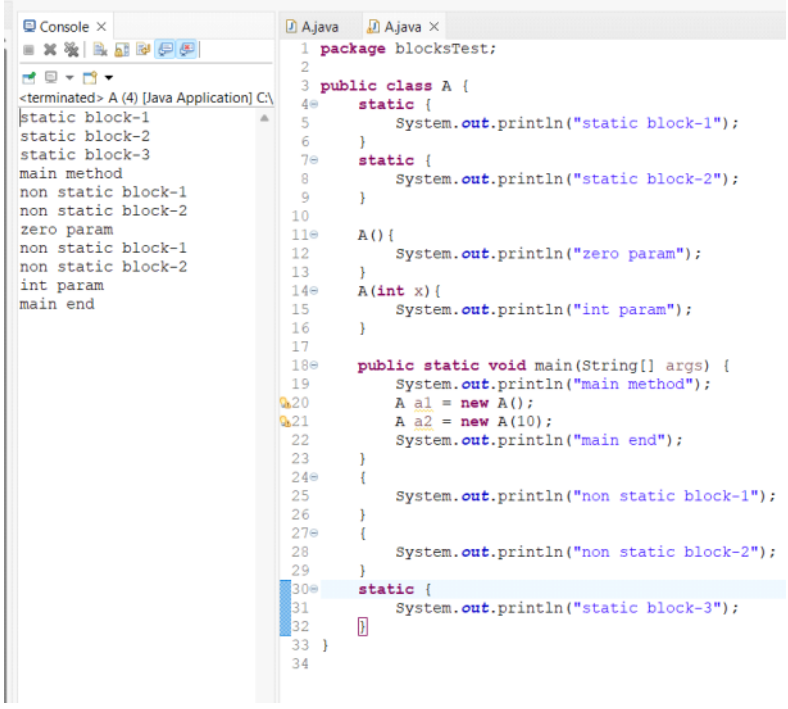
Blocks

27 February 2023 08:27

Static	**Non-Static**
Static blocks will be executed only once at the time of class loading process	Non-Static blocks will be executed whenever we create and object
Static blocks will be executed before main method	Non-Static blocks will be executed only if we create an object

{
//statements
}
	static {
	//statements
	}

	non-static blocks will be called before a constructor



File Input Stream

28 February 2023 06:25

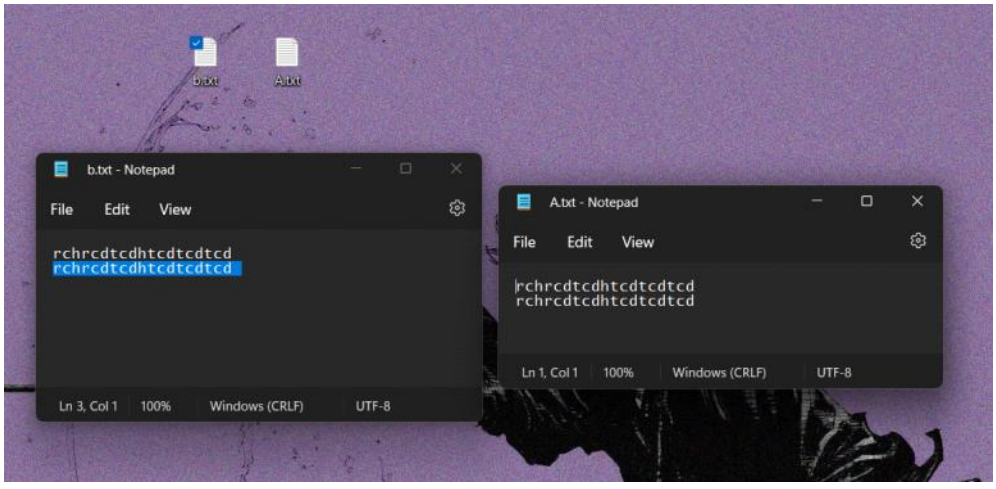
```
Console ×
<terminated> FileHandlingTest [Java Application] C:\Prograı
To be or not to be is the question
```

```java
package fileInputStream;

import java.io.File;
import java.io.FileInputStream;

public class FileHandlingTest {

    public static void main(String[] args)throws Exception {
        File f = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\Sample.txt");
        FileInputStream fis = new FileInputStream(f);
        int i = fis.read();
        StringBuffer s = new StringBuffer();
        while(i != -1) {
            s.append((char)i);
            i=fis.read();
        }
        System.out.println(s);
    }

}
```

Q. WAP to read a string from a text file and write that string into another file

```
Console ×
<terminated> FileHandlingTest [Java Appli
rchrcdtcdhtcdtcdtcd
rchrcdtcdhtcdtcdtcd
```

```java
package fileInputStream;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class FileHandlingTest {

    public static void main(String[] args)throws Exception {
        File f = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\A.txt");
        FileInputStream fis = new FileInputStream(f);
        int i = fis.read();
        StringBuffer s = new StringBuffer();
        while(i != -1) {
            s.append((char)i);
            i=fis.read();
        }
        String s1=s.toString();
        System.out.println(s1);
        File f1 = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\b.txt");
        FileOutputStream fos = new FileOutputStream(f1,true);
        for(int j=0; j<s.length(); j++) {
            char c= s1.charAt(j);
            fos.write(c);
        }


    }

}
```



Q. wap to read a file line by line by using scanner class and print it

FileHandlingTest.java

```java
package fileInputStream;

import java.io.File;

public class FileHandlingTest {

    public static void main(String[] args)throws Exception {
        File f1 = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\A.txt");
        File f2 = new File("C:\\Users\\samwa\\OneDrive\\Desktop\\B.txt");
        Scanner sc = new Scanner(f1);
        String out = "";
        while (sc.hasNextLine()) {
            String s = sc.nextLine();
            out = out + s;
            out = out + (char)10;
        }
        byte[] b = out.getBytes();
        FileOutputStream fos = new FileOutputStream(f2);
        fos.write(b);
        System.out.println("done");
```

Threads

01 March 2023 07:38

Q. What is multiprocessing?
Q. What is multitasking?
Q. What is a thread?
- It is a class in java.lang package
- by default all the programs in java will be executed in the main thread.
- therefore to create user defined threads we have two approaches
    1. implementing the runnable interface
    2. extending Thread class

Implementing runnable interface

Approach one implements Runnable interface

MyThread.java

```java
package multiThreading;

public class MyThread implements Runnable {

    public void run() {
        for(int i = 0 ; i <= 5 ; i++) {
        System.out.println(i + " ");
    }
    System.out.println("Thread finished its Task");

    }
}
```

Console

```
<terminated> ThreadDriver [Java Application] C:\
0
1
2
3
4
5
Thread finished its Task
```

ThreadDriver.java

```java
package multiThreading;

public class ThreadDriver {
public static void main(String[] args) {

        MyThread myThread = new MyThread();

        Thread thread = new Thread(myThread);

        thread.start();

}
}
```

Approach 2 extending the thread class :

MyThread.java | MyNewThread.java

```java
package multiThreading;
//app-2 extends the thread class directly

public class MyNewThread extends Thread{

    public void run() {
        for(int i = 10 ; i >= 0 ; i--) {
            System.out.println(i + " ");
        }
        System.out.println("Thread executed");
    }

}
```

ThreadDriver.java

```java
package multiThreading;

public class ThreadDriver {
public static void main(String[] args) {

        MyThread myThread = new MyThread();

        Thread thread = new Thread(myThread);

        thread.start();

        MyNewThread myNewThread = new MyNewThread();

        myNewThread.start();

}
}
```

Console

```
<terminated> ThreadDriver [Java Application] C:
0
1
2
3
4
5
Thread finished its Task
10
9
8
7
6
5
4
3
2
1
0
Thread executed
```

Q. Why does main method gets executed first?

- when JVM starts executing a java code it releases three threads
    1. Main Thread
    2. Thread Scheduler
    3. Garbage Collector

Garbage collector : to remove any unused references/objects

Properties of a Thread

01 March 2023 07:57

Properties of a Thread

1. Name
2. Id
3. Priority

Name :

- Every thread has a unique name which is assigned by the programmer to identify the threads uniquely.

Id :

- The Id of the thread will be assigned by the thread scheduler and hence it cannot be changed by the programmer

Priority :

- The priority of the thread helps the thread scheduler to order the execution of the threads
- The priority of the thread can be changed by using setPriority method
- The priority of the thread should always be within 1 - 10 , where 1 is the lowest priority and 10 is the highest and 5 is normal priority

FileHandlingTest.java | FileHandlingTest2.java | ThreadEx1.java

```java
package multiThreading;

public class ThreadEx1 {

    public static void main(String[] args) {
        Thread thread = Thread.currentThread();
        System.out.println(thread.getId());
        System.out.println(thread.getName());
        System.out.println(thread.getPriority());

    }

}
```

Console

```
<terminated> Thre
1
main
5
```

Life Cycle of a Thread

03 March 2023 07:55