Day 10

- In java, checking array bounds is a job of JVM.
- Using illegal index, if we try to access element from array then JVM throws ArrayIndexOutOfBoundsException.

```
int[] arr = new int[ ] { 10, 20, 30 };
//int element = arr[ -1  ]; //ArrayIndexOutOfBoundsException
int element = arr[ arr.length  ];//ArrayIndexOutOfBoundsException
```

- foreach loop is also called as iterator in java.
- It is used to traverse collection in forward direction only. During travsering, we can only read the values.

Array of value type

```
boolean[] arr = new boolean[ 3 ];
```

- If we create array of value type then default value of array depends on default value of data type.

Array of references

```
//Single dynamic object in C++
Complex *ptr = new Complex( );
//Array of objects in C++
Complex *ptr = new Complex[ 3 ];
```

```
Complex c1;
//c1 is object reference / reference
Complex c1 = new Complex();
//It is instantiation of single java instance

Complex[] arr;
//arr is reference of single dimensional array

Complex[] arr = new Complex[ 3 ];
//It is array of references in java whose default value is null.
```

Array of instances

```
Complex c1 = new Complex();
Complex c2 = new Complex();
```

```
Complex c3 = new Complex();
```

```
Complex[] arr = new Complex[ 3 ];
arr[ 0 ] = new Complex();
arr[ 1 ] = new Complex();
arr[ 2 ] = new Complex();
```

```
Complex[] arr = new Complex[ 3 ];
for( int i = 0; i < arr.length; ++ i )
    arr[ i ] = new Complex();
```

Passing argument by reference:

- In java, we can pass argument to the method by value only.
- Using array, we can pass, argument to the method by reference.

```
private static void swap(int[] arr)
{
    int temp = arr[ 0 ];
    arr[ 0 ] = arr[ 1 ];
    arr[ 1 ] = temp;
}
public static void main(String[] args)
{
    int a = 10;
    int b = 20;

    int[] arr = new int[ ] { a, b };
    Program.swap( arr );
    a = arr[ 0 ]; b = arr[ 1 ];

    System.out.println("a    :    "+a);
    System.out.println("b    :    "+b);
}
```

Variable argument method

```
private static void sum( int... args )
{
    int result = 0;
    for( int element : args )
        result = result + element;
    System.out.println("Result    :    "+result);
}
```

- public static String format(String format, Object... args);
- public PrintStream printf(String format, Object... args);
- public Object invoke(Object obj, Object... args);

System Date and Time

- Using Calendar:
  - Calendar is abstract class declared in java.util package.
  - Fields:
    1. public static final int DATE
    2. public static final int MONTH
    3. public static final int YEAR
    4. public static final int HOUR
    5. public static final int MINUTE
    6. public static final int SECOND
  - Methods
    1. public static Calendar getInstance()
    2. public int get(int field)

```
Calendar c = Calendar.getInstance();
int day = c.get(Calendar.DATE);
int month = c.get(Calendar.MONTH) + 1;
int year = c.get(Calendar.YEAR);
```

- Using Date:
  - It is a concrete class declared in java.util package.
  - It is Deprecated class.
  - If we want to format Date and Time then we should use SimpleDateFormat class which is declared in java.text package.

```
//Date date = new Date(119, 10, 5);
Date date = new Date();
String pattern = "dd/MM/yyyy";
SimpleDateFormat sdf = new SimpleDateFormat(pattern);
String strDate = sdf.format(date);
```

- Using LocalDate:
  - It is a final class declared in java.time package.

```
LocalDate ld = LocalDate.now();
int day = ld.getDayOfMonth();
int month = ld.getMonthValue();
int year = ld.getYear();
```

Hierarchy

- It is a major pillar of oops.
- Level/Order/Ranking of abtrsction is called hierarchy.
- If we want to achieve reusability then we should use hierarchy.
- Types hierarchies:

1. Has-a - Association
2. Is-a - Inheritance
3. Use-a - Dependancy
4. Creates-a - Instantiation

Association

- Example:
    1. Car has-a engine
    2. Room has-a chair
    3. Employee has-a joinDate
- If has-a relationship exist between the types then we should use association.

```
//Car has-a Engine or Engine is a part of Car
class Engine
{   }
class Car
{
    Engine e = new Engine();    //Association
}
Car c = new Car();
```

- If instance is a part of another instance then it is called association.
- In java, association do not represent physical containment.

```
class Date{ }
class Address{  }
class Employee
{
    private String name;    //Association
    private int empid;
    private float salary;
    private Date joinDate;  //Association
    private Address currentAddress; //Association
}
```

Inheritance

```
class Person    //Parent/Super class
{   }
class Employee extends Person   //Child/Sub class
{   }
```

- using extends keyword,we can create sub class.
- During inheritace all the fields of super class inherit into sub class but only non static field get space inside instance.
- Except constructor, all the methods of super class inherit into sub class.