Day 2

Console IO

Console

```
- Keyborad and monitor is collectivly called as      console.
- Console Input -> Keyboard
- Console Output -> Monitor
```

Stream

It is an abstraction(object) that is used to produce(write) and consume(read) information from source to destination. Stream is always associated with physical device.

Standard Stream Objects of C associated with console

```
1. stdin
2. stdout
3. stderr
```

Standard Stream Objects of C++ associated with console

```
1. cin
2. cout
3. cerr
4. clog
```

Standard Stream Objects of java associated with console

```
1. System.in -> Associated with Keyboard
2. System.out-> Associated with Monitor
3. System.err-> Associated with Monitor( Error Stream)
```

"public static Console console()"

It is method of java.lang.System class.

- If we want to access any type outside package then either we should use F.Q.TypeName or import statement. e.g java.io.Console c = ...;

  import java.io.Console; Console c = ...;

```
import java.io.Console;
class Program
{
    public static void main(String[] args)
    {
        //java.io.Console console =  System.console();
        Console console =  System.console();
        System.out.print("Name  :   ");
        String name = console.readLine();
        System.out.print("Empid  :   ");
        int empid= Integer.parseInt( console.readLine() );
        System.out.print("Salary  :   ");
        float salary = Float.parseFloat(console.readLine());
        System.out.printf("%-30s%-5d%-10.2f\n",name, empid, salary);
    }
}
```

java.util.Scanner class

```
- It is a final class declared in java.util package.
- If we want to perform input operation on Console/File etc then we should
use Scanner class.
- Following are Method's of Scanner class
    1. public String nextLine()
    2. public int nextInt()
    3. public float nextFloat()
    4. public double nextDouble()
```

```
     public static void main( String[] args )
    {
        Scanner sc = new Scanner( System.in);
        System.out.print("Name  :   ");
        String name = sc.nextLine();
        System.out.print("Empid  :   ");
        int empid= sc.nextInt();
        System.out.print("Salary  :   ");
        float salary = sc.nextFloat();
        System.out.printf("%-30s%-5d%-10.2f\n",name, empid, salary);
    }
```

java.io.BufferedReader

```
- It is a class declared in java.io.
```

```
    public static void main(String[] args) throws IOException
    {
        BufferedReader reader = new BufferedReader( new InputStreamReader(
System.in) );
        System.out.print("Name  :   ");
        String name = reader.readLine();
        System.out.print("Empid  :   ");
        int empid= Integer.parseInt( reader.readLine() );
        System.out.print("Salary  :   ");
        float salary = Float.parseFloat(reader.readLine());
        System.out.printf("%-30s%-5d%-10.2f\n",name, empid, salary);
    }
```

Class

```
- It is collection of fields and methods
- Structure and behavior of instance depends on class hence     class is
considered as a template/model/blueprint for instance
- Class represents set of instances that share common structure and common
behavior
- e.g Laptop
```

Object

```
- It is variable or instance of class
- Any entity which has physical existance is called object.
- Any entity which has state, behavior and identity is called object
- e.g MacBook Air
```

Characteristics of object

```
- State : value stored inside object is called state
 Value of the field represents state of instance.
- Behavior : Operations that we can perform on instance is called behavior
of instance.
- Identity : value of any field that is used to identify instance uniquely
is called identity.
```

Access Modifier

```
- If we want to control visibility of members of the class then we should
use access modifier.
1. private( - )
2. package level private / default ( ~ )
3. protected( # )
4. public( + )
```

- Class can contain static as well as non static fields. Non static fields get space once per instance whereas static fields get space once per class in other word, all the instances of same class share single copy of static fields.
- Methods do not get space inside instance.
- If we want to create instance of a class(reference type) then it is mandatory to use new operator.
- Everything on heap section is anonymous
- If we want to perform operations on instance then we should create "object reference"/reference.
- Process of creating instance from class is called instantiation.
- If we want to process state of instance then we should call method on instance. This process of calling method on instance is called "Message Passing".
- Method is also called as operation, behavior, message etc.
- "this" is implict reference variable available in every non static method of a class that is used to store reference of current instance.
- Using "this", field and method can communicate with each other hence it is considered as a link or connection between it.
- null is literal in java which is used to initialize reference variable.
- If reference contain null value then it is called null reference variable or null object.

```
import java.util.Scanner;
class Complex
{
    //Fields
    private int real;   //Instance Variable
    private int imag;   //Instance Variable
    //Compex this = c1
    public int getReal( )
    {
        return this.real;
    }
    //Compex this = c1
    public void setReal( int real )
    {
        this.real = real;
    }
    //Compex this = c1
    public int getImag( )
    {
        return this.imag;
    }
    //Compex this = c1
```

```
    public void setImag( int imag )
    {
        this.imag = imag;
    }
    public void acceptRecord( )
    {
        Scanner sc  =new Scanner(System.in);
        System.out.print("Real Number   :   ");
        this.real = sc.nextInt();
        System.out.print("Imag Number   :   ");
        this.imag = sc.nextInt();
    }
    //Complex this=c1
    public void printRecord( )
    {
        System.out.println("Real Number :   "+this.real);
        System.out.println("Imag Number :   "+this.imag);
    }
}
class Program
{
    public static void main(String[] args)
    {
        Complex c1 = null;
        //c1.printRecord( );  //NullPointerException
        c1 = new Complex();
        c1.printRecord();
    }
    public static void main2( String[] args )
    {
        Complex c1 = new Complex();
        c1.acceptRecord( );
        c1.printRecord( );
    }
    public static void main1( String[] args )
    {
        Complex c1 = new Complex(); //Instantiation

        c1.setReal( 10 );
        c1.setImag( 20 );

        int real = c1.getReal( );
        System.out.println("Real    :   "+real);
        int imag = c1.getImag();

        System.out.println("Real    :   "+imag);
    }
}
```