

Q1. Explain Command Line Argument in java Programming?

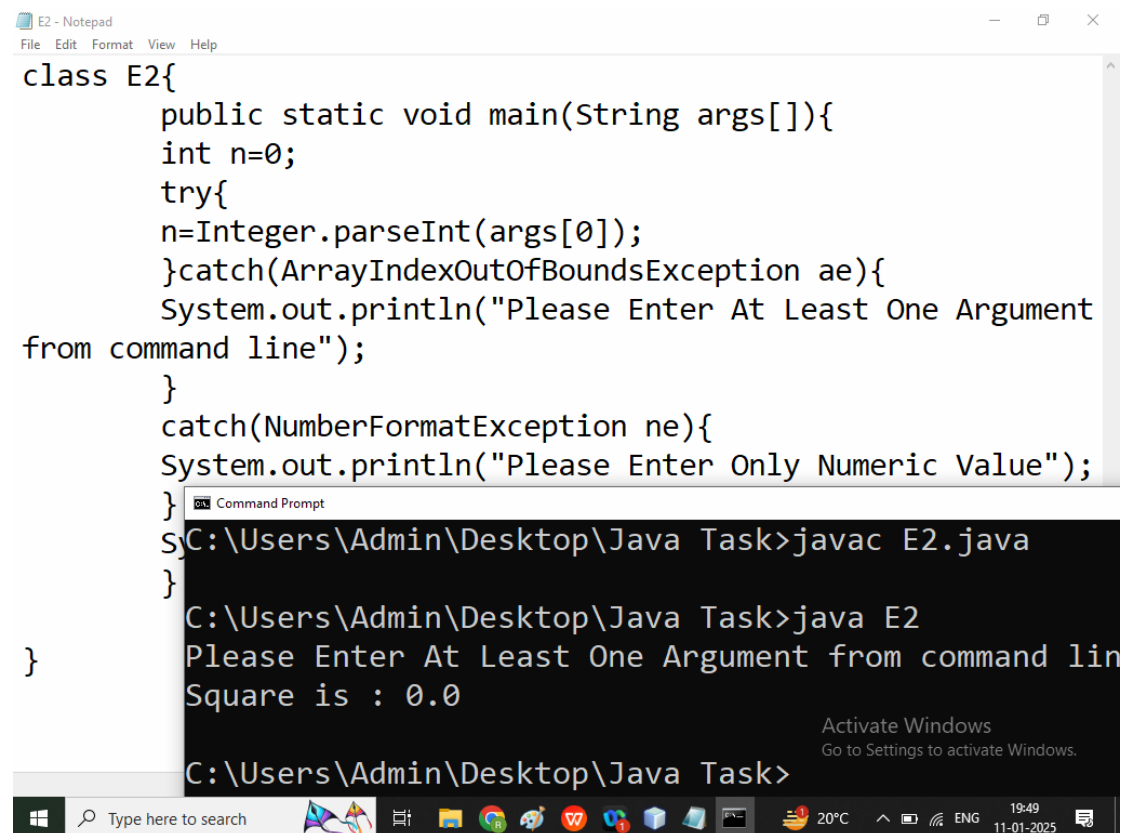
Ans: Command Line Argument is used to allow to pass information to a program at the time of execution of the Program

Syntax:

java ClassName Hello Hi How Are You

java ClassName 1 2 3 4 5

Java ClassName



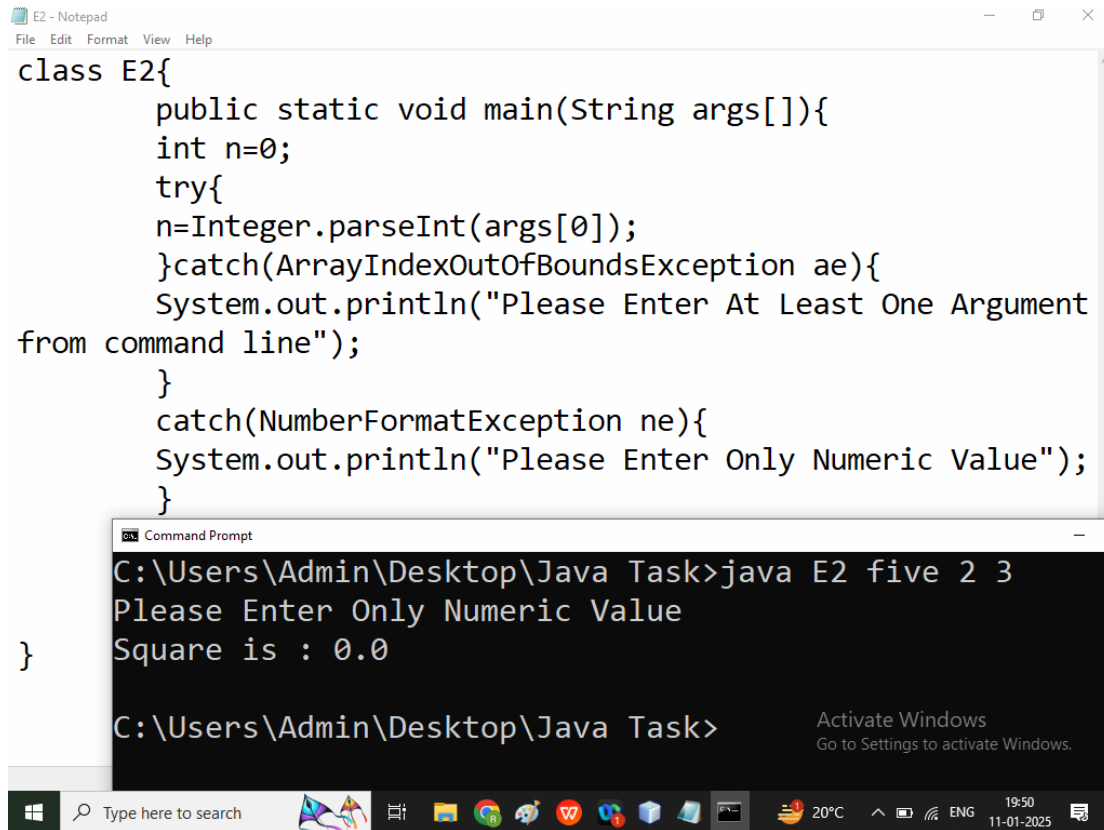
The screenshot displays a Windows desktop environment. In the foreground, a Notepad window titled 'E2 - Notepad' contains the following Java code:

```
class E2{
    public static void main(String args[]){
        int n=0;
        try{
            n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
            System.out.println("Please Enter At Least One Argument
from command line");
        }
        catch(NumberFormatException ne){
            System.out.println("Please Enter Only Numeric Value");
        }
    }
}
```

Overlaid on the bottom right of the Notepad window is a Command Prompt window. It shows the following commands and output:

```
C:\Users\Admin\Desktop\Java Task>javac E2.java
C:\Users\Admin\Desktop\Java Task>java E2
Please Enter At Least One Argument from command lin
Square is : 0.0
```

The Command Prompt window also shows a Windows activation watermark: 'Activate Windows Go to Settings to activate Windows.' The Windows taskbar at the bottom includes the Start button, a search bar, and various application icons. The system tray on the right shows the temperature (20°C), time (19:49), and date (11-01-2025).



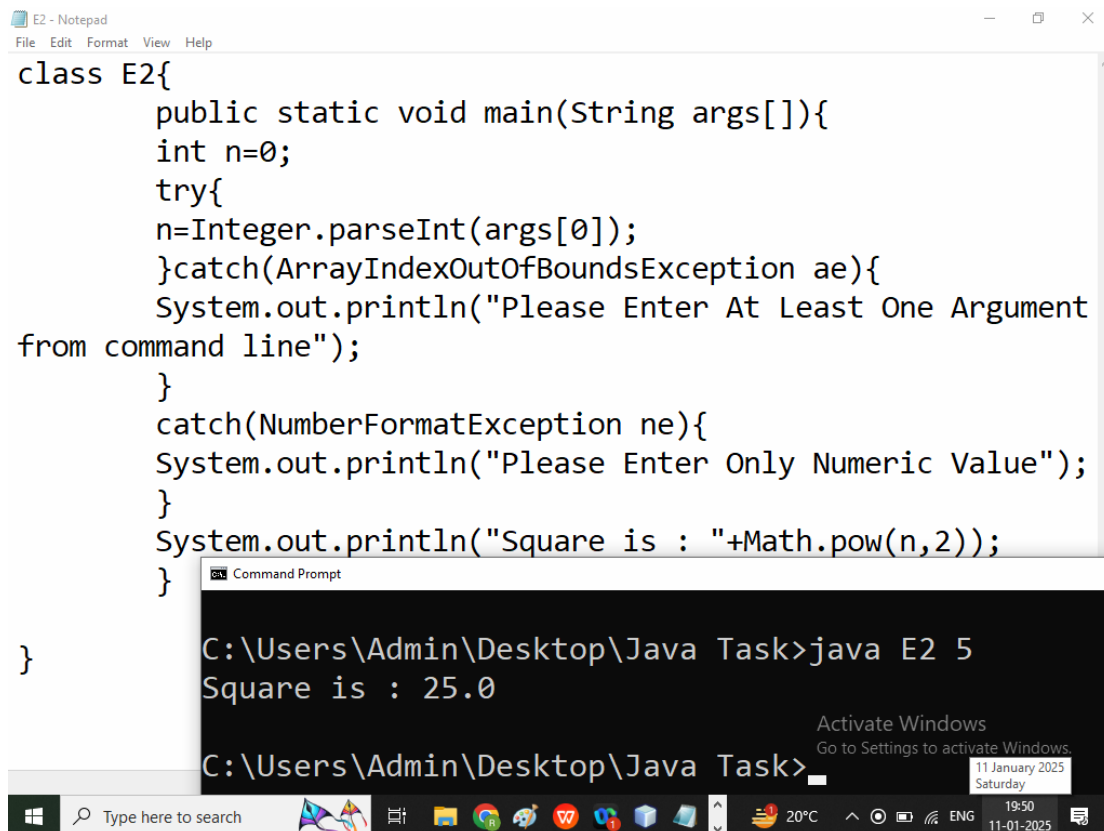
The screenshot shows a Windows desktop environment. A Notepad window titled 'E2 - Notepad' contains the following Java code:

```
class E2{
    public static void main(String args[]){
        int n=0;
        try{
            n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
            System.out.println("Please Enter At Least One Argument
from command line");
        }
        catch(NumberFormatException ne){
            System.out.println("Please Enter Only Numeric Value");
        }
    }
}
```

Overlaid on the bottom right of the Notepad window is a Command Prompt window titled 'Command Prompt'. It shows the command `java E2 five 2 3` being executed. The output of the program is displayed as follows:

```
C:\Users\Admin\Desktop\Java Task>java E2 five 2 3
Please Enter Only Numeric Value
Square is : 0.0
C:\Users\Admin\Desktop\Java Task>
```

The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with a temperature of 20°C and the date 11-01-2025.



This screenshot is similar to the first one, but the Command Prompt window shows a different execution. The Notepad window contains the same Java code as before, but with an additional line of code at the end of the `main` method:

```
System.out.println("Square is : "+Math.pow(n,2));
```

The Command Prompt window shows the command `java E2 5` being executed. The output is:

```
C:\Users\Admin\Desktop\Java Task>java E2 5
Square is : 25.0
C:\Users\Admin\Desktop\Java Task>
```

The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with a temperature of 20°C and the date 11-01-2025. A date picker is visible in the system tray, showing '11 January 2025 Saturday'.

```
class E2{
    public static void main(String args[]){
        int n=0;
        try{
```

```

        n=Integer.parseInt(args[0]);
    }catch(ArrayIndexOutOfBoundsException ae){
        System.out.println("Please Enter At Least One Argument from command line");
    }
    catch(NumberFormatException ne){
        System.out.println("Please Enter Only Numeric Value");
    }
    System.out.println("Square is : "+Math.pow(n,2));
}

}

}



---


class E2{
    public static void main(String args[]){
        int n=0;
        try{
            n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
            System.out.println("Please Enter At Least One Argument from command line");
            ae.printStackTrace();

        }
        catch(NumberFormatException ne){
            System.out.println("Please Enter Only Numeric Value");
            ne.printStackTrace();

        }
        System.out.println("Square is : "+Math.pow(n,2));
    }
}

}

}

```

Q1. Explain User Defined Exception class or Customize Exception class in java Programming?

Ans: In java if you are not satisfied from the pre-defined exception then you also create your own Exception class/ User Defined Exception/ Customize Exception

You can create a User defined Exception or Customize Exception by extending Throwable or Exception Class

Steps to create a User Defined Class in java

1. Create a Custom Class/ User Defined Class for the exception by extending Throwable or Exception(define Member data String)
2. Define a Parameterized Constructor and method to return message
3. Throw the Exception from where you handle it.
4. Handle The Exception using try catch

```
class NegativeException extends Throwable{  
    private String msg;  
    public NegativeException(String msg){  
        this.msg=msg;  
    }  
}
```

```
    public String getMsg(){  
        return msg;  
    }  
}
```

```
}
```

```
class E2{  
    public static void main(String args[]){  
        int n=0;  
        try{  
            n=Integer.parseInt(args[0]);  
            if(n<0){
```

```
        NegativeException x=new
NegativeException("Negative Number
Exception Occur");
        throw x;
    }
    }catch(ArrayIndexOutOfBoundsException
ae){
        System.out.println("Please Enter At Least
One Argument from command line");
        ae.printStackTrace();

    }
    catch(NumberFormatException ne){
        System.out.println("Please Enter Only
Numeric Value");
        ne.printStackTrace();

    }
    catch(NegativeException y){
        System.out.println(y.getMsg());
    }
    System.out.println("Square is :
"+Math.pow(n,2));
}
```

}

