QB:-88:- Write a method for computing x ^ y by doing repetitive multiplication. x and y are of type integer and are to be given as command line arguments. Raise and handle exception(s) for invalid values of x and y. Also define method main.

```
public class PowerCalculator {

    public static void power(int x, int y) {

        double result = 1;
        if (y > 0) {
            for (int i = 1; i <= y; i++) {
                result *= x;
                System.out.println(x + "^" + y + " = " + result);
            }
        } else if (y == 0) {
            System.out.println(x + "^" + y + " = " + 1);
        } else {
            for (int i = 1; i <= (-y); i++) {
                result *= x;
                System.out.println(x + "^" + y + " = " + (1 / result));
            }
        }
    }

    public static void main(String[] args) {
        try {
            int x = Integer.parseInt(args[0]);
            int y = Integer.parseInt(args[1]);
            power(x, y);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Must Enetr 2 arguments only");
        } catch (NumberFormatException e) {
            System.out.println("Error: Enter  integer input only");
        }
    }
}
/*PS F:\Exception\LJ> javac PowerCalculator.java
PS F:\Exception\LJ> java PowerCalculator
Must Enetr 2 arguments only
PS F:\Exception\LJ> java PowerCalculator 2 0
2^0 = 1
PS F:\Exception\LJ> java PowerCalculator 2 1
2^1 = 2.0
PS F:\Exception\LJ> java PowerCalculator 2 -1
2^-1 = 0.5
PS F:\Exception\LJ> java PowerCalculator 2 one
Error: Enter  integer input only */
```

QB:- 89:-Write an application that searches through its command-line argument. If an argument is found that does not begin with and upper case letter, display error message and terminate.

```
public class ArgumentChecker {

    public static void checkArguments(String[] args) {
        for (String arg : args) {
            if (!(arg.charAt(0) >= 'A' && arg.charAt(0) <= 'Z')) {
                throw new IllegalArgumentException("Invalid argument: " +
arg);
            }
        }

    }

    public static void main(String[] args) {
        try {
            if (args.length == 0) {
                System.out.println("Enter atleast one Arguments");
            } else {
                checkArguments(args);
                System.out.println("All arguments are valid.");
            }
        } catch (IllegalArgumentException e) {
            System.err.println(e.getMessage());
        }
    }
}
/*PS F:\Exception\LJ> javac ArgumentChecker.java
PS F:\Exception\LJ> java ArgumentChecker
Enter atleast one Arguments
PS F:\Exception\LJ> java ArgumentChecker hii
Invalid argument: hii
PS F:\Exception\LJ> java ArgumentChecker Hii How
All arguments are valid. */
```

QB:-90:- Write a program to create user define exception MyException. Define a class ExceptionDemo, that has a method named compute(int a) which throws a MyException object, when compute(int a)'s integer parameter is greater than 10

```
import java.util.Scanner;

class MyException extends RuntimeException {

}
```

```
class ExceptionDemo {
    public void compute(int a) throws MyException {
        if (a > 10) {
            throw new MyException();
        } else {
          System.out.println("Integer parameter is less than or equal to 10");
        }
    }
}
public class Main {
    public static void main(String[] args) {
        ExceptionDemo d = new ExceptionDemo();
        try {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter any Integer Number: ");
            int a = sc.nextInt();
            d.compute(a);
        } catch (MyException e) {
            System.out.println("Integer parameter is greater than 10");
        }
    }
}
```

QB:-92:-Write a complete program to accept N integer numbers from the command line. Raise and handle exceptions for following cases :

- when a number is –ve

- when a number is evenly divisible by 10

- when a number is greater than 1000 and less than 2000

- when a number is greater than 7000

Skip the number if an exception is raised for it, otherwise add it to find total sum

```
public class NumberValidator {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 0; i < args.length; i++) {
            try {
                int num = Integer.parseInt(args[i]);
                if (num < 0) {
                    throw new IllegalArgumentException("Negative number not
allowed");
                }
                if (num % 10 == 0) {
                    throw new Exception(num+" Number is evenly divisible by
10");
                }
```

```
                if (num > 1000 && num < 2000) {
                    throw new Exception(num+" Number is greater than 1000 and
less than 2000");
                }
                if (num > 7000) {
                    throw new Exception(num+" Number is greater than 7000");
                }
                sum += num;
            } catch (NumberFormatException e) {
                System.out.println(args[i] + " is not a valid integer");
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
            } catch (Exception e) {
                System.out.println(e.getMessage() + " - Skipping number " +
args[i]);
            }
        }
        System.out.println("Total sum is: " + sum);
    }
}
```

QB:-93:-Declare a class called coordinate to represent 3 dimensional Cartesian coordinates(x, y and z).

Define following methods:

- constructor

- display, to print values of members

- add_coordinates, to add three such coordinate objects to produce a resultant

coordinate object. Generate and handle exception if x, y and z coordinates of the resultare zero.

- main, to show use of above methods.

```
public class Coordinate {
     double x;
     double y;
     double z;

    public Coordinate(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public void display() {
```

```
        System.out.println("(" + x + ", " + y + ", " + z + ")");
    }

    public static Coordinate addCoordinates(Coordinate c1, Coordinate c2,
Coordinate c3) throws Exception {
        double x = c1.x + c2.x + c3.x;
        double y = c1.y + c2.y + c3.y;
        double z = c1.z + c2.z + c3.z;
        if (x == 0 && y == 0 && z == 0) {
            throw new Exception("Resultant coordinate has zero values for x, y
and z");
        }
        return new Coordinate(x, y, z);
    }

    public static void main(String[] args) {
        Coordinate c1 = new Coordinate(1, 2, 3);
        Coordinate c2 = new Coordinate(4, 5, 6);
        Coordinate c3 = new Coordinate(7, 8, 9);
        c1.display();
        c2.display();
        c3.display();
        try {
            Coordinate result = Coordinate.addCoordinates(c1, c2, c3);
            result.display();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
/*when we will put all the values 0 Exception occurs*/
```

QB:-95:- Write a program to create user define exception MyException. Define a class ExceptionDemo that has a method named compute( ) which throws a MyException object, when compute( )'s integer parameter is divisible by 7 and not divisible by 5.

```
import java.util.Scanner;

class MyException extends RuntimeException {
    public MyException() {
        super("Number is divisible by 7 but not divisible by 5");
    }
}

public class ExceptionDemo {
    public void compute(int num) throws MyException {
        if (num % 7 == 0 && num % 5 != 0) {
            throw new MyException();
```

```
        }
        System.out.println("Number is valid");
    }

    public static void main(String[] args) {
        ExceptionDemo d = new ExceptionDemo();
        try {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter any Integer Number: ");
            int a = sc.nextInt();
            d.compute(a);
        } catch (MyException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
/*PS F:\Exception\LJ> javac ExceptionDemo.java
PS F:\Exception\LJ> java ExceptionDemo
Enter any Integer Number: 21
Error: Number is divisible by 7 but not divisible by 5
PS F:\Exception\LJ> java ExceptionDemo
Enter any Integer Number: 35
Number is valid */
```

QB:-96:-Write an application that searches through its command-line argument. If any of command line argument is found nagative then display error message.

```
public class QB96 {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Enter atleast one Argument: ");
        } else {
            for (String arg : args) {
                try {
                    int num = Integer.parseInt(arg);
                    if (num < 0) {
                  throw new IllegalArgumentException("Negative number found");
                    }
                } catch (NumberFormatException e) {
                    // Ignore non-integer arguments
                } catch (IllegalArgumentException e) {
                    System.out.println("Error: " + e.getMessage());
                    return;
                }
            }
            System.out.println("All arguments are valid");
        }
```

```
    }
}
/*PS F:\Exception\LJ> javac QB96.java
PS F:\Exception\LJ> java QB96
Enter atleast one Argument:
PS F:\Exception\LJ> java QB96 12 1 one
All arguments are valid
PS F:\Exception\LJ> java QB96 12 5 -66
Error: Negative number found */
```

QB:-97:-Write an application that searches through its command-line argument. If first command line argument is found zero then display error message.

```
public class QB97 {
    public static void main(String[] args) {
        if (args.length > 0) {
            try {
                int num = Integer.parseInt(args[0]);
                if (num == 0) {
                    throw new IllegalArgumentException("First argument cannot
be zero");
                }
            } catch (NumberFormatException e) {
                System.out.println("Error: First argument must be an
integer");
                return;
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("Error: No arguments provided");
            return;
        }
        System.out.println("All arguments are valid");
    }
}
/*PS F:\Exception\LJ> javac QB97.java
PS F:\Exception\LJ> java QB97
Error: No arguments provided
PS F:\Exception\LJ> java QB97 one
Error: First argument must be an integer
PS F:\Exception\LJ> java QB97 12
All arguments are valid
PS F:\Exception\LJ> java QB97 012
All arguments are valid
PS F:\Exception\LJ> java QB97 0 123
```

```
Error: First argument cannot be zero */
```

QB:-98:-Write an application that searches through its command-line argument. If sum of first and second command line argument is found 10 then display error message.

```
public class QB98 {
    public static void main(String[] args) {
        if (args.length >= 2) {
            try {
                int num1 = Integer.parseInt(args[0]);
                int num2 = Integer.parseInt(args[1]);
                if (num1 + num2 == 10) {
                    throw new IllegalArgumentException("Sum of first two
arguments cannot be 10");
                }
            } catch (NumberFormatException e) {
                System.out.println("Error: Arguments must be integers");
                return;
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("Error: At least two arguments must be
provided");
            return;
        }
        System.out.println("All arguments are valid");
    }
}
/*PS F:\Exception\LJ> javac QB98.java
PS F:\Exception\LJ> java QB98
Error: At least two arguments must be provided
PS F:\Exception\LJ> java QB98 one 10
Error: Arguments must be integers
PS F:\Exception\LJ> java QB98 1 5
All arguments are valid
PS F:\Exception\LJ> java QB98 1 9
Error: Sum of first two arguments cannot be 10
PS F:\Exception\LJ> java QB98 0 10
Error: Sum of first two arguments cannot be 10 */
```

QB:-99:- write a program to enter the five subject marks out of 100 of student. If any subject marks is less than 35 then programm will generate exception

```
import java.util.Scanner;

class SubjectMarksException extends Exception {
```

```java
    public SubjectMarksException(String message) {
        super(message);
    }
}

public class QB99 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] marks = new int[5];
        try {
            for (int i = 0; i < 5; i++) {
                System.out.print("Enter marks for subject " + (i + 1) + ": ");
                marks[i] = sc.nextInt();
                if (marks[i] < 35) {
                    throw new SubjectMarksException("Marks cannot be less than
35");
                }
            }
            System.out.println("All marks are valid");
        } catch (SubjectMarksException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
/*PS F:\Exception\LJ> javac QB99.java
PS F:\Exception\LJ> java QB99
Enter marks for subject 1: 40
Enter marks for subject 2: 36
Enter marks for subject 3: 65
Enter marks for subject 4: 41
Enter marks for subject 5: 45
All marks are valid
PS F:\Exception\LJ> java QB99
Enter marks for subject 1: 10
Exception: Marks cannot be less than 35 */
```