Hands On - Day 08 - Exception Handling

1. Question 1:

Problem Statement:

Write a Java program to get the citizen details like aadhar number, name, city, state and country. Define an custom exception called “NoMatchExcepion”, that is throwr when the country name is not equal to “India”.

Analysis:

1. Create a class called Citizen with attributes aadhar number, name, city, state and country.
2. Create a custom exception called NoMatchException.
3. Create a method called validateCountry() to check the country name.
4. If the country name is not equal to “India”, throw NoMatchException.
5. Create an object of Citizen class and call the validateCountry() method.
6. Handle the exception using try-catch block.

Code:

```java
class Citizen {
    String aadharNumber, name, city, state, country;
    Citizen(String aadharNumber, String name, String city, String state, String country) {
        this.aadharNumber = aadharNumber;
        this.name = name;
        this.city = city;
        this.state = state;
        this.country = country;
    }
    void validateCountry() throws NoMatchException {
        if(!country.equals("India")) {
            throw new NoMatchException("Country name is not equal to India");
        }
    }
}

class NoMatchException extends Exception {
    NoMatchException(String message) {
        super(message);
    }
}

public class Main {
    puublic class Main {
    public static void main(String[] args) {
        Citizen citizen1 = new Citizen("123456789012", "Shabari", "Salem", "Tamil Nadu", "India");
        Citizen citizen2 = new Citizen("123456789012", "Sandy", "Jeru Salem", "Canada", "USA");
        try {
            citizen1.validateCountry();
            citizen2.validateCountry();
        } catch (NoMatchException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output:

```
Country name is not equal to India
```

---

2. Question 2:

Problem Statement:

Create a calculator Application which does basic operations like addition, subtraction, multiplication and division with the following criteria

1. Input must be in number format. No character or string inputs are not accepted.
2. For division, denominator should not be zero.
3. For multiplication, multiplier and multiplicand should not be 0 or 1.

Create user defined exceptions wherever it is necessary and handle it appropriately.

Analysis:

1. Create a class called Calculator with methods add(), subtract(), multiply() and divide().
2. Create a custom exception called InvalidInputException.
3. Create a custom exception called InvalidDivisionException.
4. Create a custom exception called InvalidMultiplicationException.
5. Create an object of Calculator class and call the methods.
6. Handle the exception using try-catch block.

Code:

```
class Calculator {
    void add(int a , int b){
        System.out.println("Addition: "+ (a+b));
    }

    void sub(int a, int b){
        System.out.println("Subtraction: "+ (a-b));
    }

    void multiply(int a, int b) throws InvalidMultiplicationException {
        if(a == 0 || b == 0 || a == 1 || b == 1) {
            throw new InvalidMultiplicationException("Multiplication is not possible");
        }
        System.out.println("Multiplication: " + (a * b));
    }
    void divide(int a, int b) throws InvalidDivisionException {
        if(b == 0) {
            throw new InvalidDivisionException("Division is not by 0.");
        }
        System.out.println("Division: " + (a / b));
    }
}


class InvalidInputException extends Exception {
    InvalidInputException(String message) {
        super(message);
    }
}

class InvalidDivisionException extends Exception {
    InvalidDivisionException(String message) {
        super(message);
    }
}

class InvalidMultiplicationException extends Exception {
    InvalidMultiplicationException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        try {
            calculator.add(10, 20);
            calculator.subtract(20, 10);
            calculator.multiply(0, 5);
            calculator.divide(10, 0);
        } catch (InvalidInputException | InvalidDivisionException | InvalidMultiplicationException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output:

```
Addition: 30
Subtraction: 10
Multiplication is not possible
```

---

3. Question 3:

Problem Statement:

Create a class MyCalculator which consists of a single method long power(int, int). This method takes two integers, n and p ,pass parameters and finds n^p. If either n o
p is negative, then the method must throw an exception which says " n or p should not negative". Also, if both n and p are zero, then the method must throw an exception which says "n and p should not be Zero ".

Sample Input

```
3 5
2 4
0 0
-1 -2
-1 3
```

Sample Output

```
243
16
java.lang.Exception: n and p should not be zero.
java.lang.Exception: n or p should not be negative.
java.lang.Exception: n or p should not be negative.
```

Analysis:

1. Create a class called MyCalculator with a method power().
2. Create a custom exception called InvalidInputException.
3. If either n or p is negative, throw an exception.
4. If both n and p are zero, throw an exception.

5. Create an object of MyCalculator class and call the power() method.
6. Handle the exception using try-catch block.

Code:

```java
import java.util.Scanner;
class MyCalculator {
    long power(int n, int p) throws InvalidInputException  {
        if(n < 0 && p < 0) {
            throw new InvalidInputException("n or p should not be negative");
        }
        if(n == 0 && p == 0) {
            throw new InvalidInputException("n and p should not be zero");
        }
        return (long) Math.pow(n, p);
    }
}

class InvalidInputException extends Exception {
    InvalidInputException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        MyCalculator myCalculator = new MyCalculator();
        Scanner scanner = new Scanner(System.in);
        int n, p;
        n = scanner.nextInt();
        p = scanner.nextInt();
        try {
            System.out.println(myCalculator.power(n, p));
        } catch (InvalidInputException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output:

```
0 0
com.hands_on.Question3.InvalidInputException: n and p should not be zero
```

---

4. Question 4:

Problem Statement:

Accept Username and password values from the User . Throw user defined exceptions – “InvalidUsernameException” or “InvalidPasswordException” if the username and password are invalid . Please check with Stored username and password for validity. If valid, print a message “Welcome ‘username’, else print a message Invalid username or password. A username is considered valid if all the following constraints are satisfied:

- The username consists of 6 to 30 characters inclusive. If the username consists of less than 6 or greater than 30 characters, then it is an invalid username.
- The username can only contain alphanumeric characters and underscores (_). Alphanumeric characters describe the character set consisting of lowercase characters [a – z], uppercase characters [A – Z], and digits [0 – 9].
- The first character of the username must be an alphabetic character, i.e., either lowercase character [a – z] or uppercase character [A – Z].

A password is considered valid if all the following constraints are satisfied:

- It contains at least one lowercase English character.

- It contains at least one uppercase English character.
- It contains at least one special character. The special characters are: !@#$%^&*()-+
- Its length is at least 8.
- It contains at least one digit.

Analysis:

1. Create a class called User with attributes username and password.
2. Create a custom exception called InvalidUsernameException.
3. Create a custom exception called InvalidPasswordException.
4. Create a method called validateUsername() to check the username.
5. Create a method called validatePassword() to check the password.
6. Create an object of User class and call the validateUsername() and validatePassword() methods.
7. Handle the exception using try-catch block.

Code:

```
class User {
    String username, password;

    User(String username, String password){
        this.username = username;
        this.password = password;
    }

    void validateUsername() throws InvalidUsernameException {
        if(username.length() < 6 || username.length() > 30) {
            throw new InvalidUsernameException("Invalid username");
        }
        if(!username.matches("^[a-zA-Z][a-zA-Z0-9_]*$")) {
            throw new InvalidUsernameException("Invalid username");
        }
    }

    void validatePassword() throws InvalidPasswordException {
        if(!password.matches("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*()-+]).{8,}$")) {
            throw new InvalidPasswordException("Invalid password");
        }
    }
}


class InvalidUsernameException extends Exception {
    InvalidUsernameException(String message) {
        super(message);
    }
}

class InvalidPasswordException extends Exception {
    InvalidPasswordException(String message) {
        super(message);
    }
}
package com.hands_on.Question4;


public class Main {
    public static void main(String[] args) {
        User user = new User("Shabari_123", "Shabari@123");
        User user2 = new User("Sandy_123", "Sandy");
        try {
            user.validateUsername();
```

```java
            user.validatePassword();
            System.out.println("Welcome " + user.username);
        } catch (InvalidUsernameException | InvalidPasswordException e) {
            System.out.println("Invalid username or password");
        }
        try {
            user2.validateUsername();
            user2.validatePassword();
            System.out.println("Welcome " + user.username);
        } catch (InvalidUsernameException | InvalidPasswordException e) {
            System.out.println("Invalid username or password");
        }
    }
}
```

Output:

```
Welcome Shabari_123
Invalid username or password
```