1. How can you implement the concept of Overloading with the object of Mobile class.

   i. Constructor Overloading
   ii. Method Overloading

2. How can you implement the concept of Overriding with the types of Mobile class.

   i. Show the example using different access modifiers.
   ii. Annotation
   iii. Final Keyword

```java
package Week4_Workshop.Overloading_Overriding;

import java.math.BigInteger;

class Mobile {
    final private String imeiNumber;
    final private String modelNumber;
    final private String screenSize;
    final private String batteryCapacity;

    public Mobile(String imeiNumber, String modelNumber, String screenSize, String batteryCapacity) {
        this.imeiNumber = imeiNumber;
        this.modelNumber = modelNumber;
        this.screenSize = screenSize;
        this.batteryCapacity = batteryCapacity;
    }

    public Mobile(int imeiNumber, int modelNumber, int screenSize, int batteryCapacity) {
        this.imeiNumber = String.valueOf(imeiNumber);
        this.modelNumber = String.valueOf(modelNumber);
        this.screenSize = String.valueOf(screenSize);
        this.batteryCapacity = String.valueOf(batteryCapacity);
    }

    public String getImeiNumber() {
        return imeiNumber;
    }

    public String getModelNumber() {
        return modelNumber;
    }

    public String getScreenSize() {
        return screenSize;
    }

    public String getBatteryCapacity() {
        return batteryCapacity;
    }

    void voice_call(BigInteger phoneNumber) {
        System.out.println("Calling the phone number: " + phoneNumber);
    }

    void voice_call(String email) {
        System.out.println("Calling the user with email: " + email);
    }

    void video_call(BigInteger phoneNumber) {
        System.out.println("Video calling the phone number: " + phoneNumber);
    }
```

```
    void video_call(String email) {
        System.out.println("Video calling the user with email: " + email);
    }

}

class Samsung extends Mobile {
    final private String cameraMegapixel;

    public Samsung(int imeiNumber, int modelNumber, int screenSize, int batteryCapacity, int cameraMegapixel) {
        super(imeiNumber, modelNumber, screenSize, batteryCapacity);
        this.cameraMegapixel = String.valueOf(cameraMegapixel);
    }

    public Samsung(String imeiNumber, String modelNumber, String screenSize, String batteryCapacity,
            String cameraMegapixel) {
        super(imeiNumber, modelNumber, screenSize, batteryCapacity);
        this.cameraMegapixel = cameraMegapixel;
    }

    public String getCameraMegapixel() {
        return cameraMegapixel;
    }
}

class Iphone extends Mobile {
    final private String iosVersion;

    public Iphone(String imeiNumber, String modelNumber, String screenSize, String batteryCapacity,
            String iosVersion) {
        super(imeiNumber, modelNumber, screenSize, batteryCapacity);
        this.iosVersion = iosVersion;
    }

    public Iphone(int imeiNumber, int modelNumber, int screenSize, int batteryCapacity,
            int iosVersion) {
        super(imeiNumber, modelNumber, screenSize, batteryCapacity);
        this.iosVersion = String.valueOf(iosVersion);
    }

    public String getIosVersion() {
        return iosVersion;
    }

    @Override
    void video_call(BigInteger phoneNumber) {
        System.out.println("FaceTiming the phone number: " + phoneNumber);
    }
```

```java
    @Override
    void video_call(String email) {
        System.out.println("FaceTiming the user with email: " + email);
    }
}

class Consumer {
    Run | Debug
    public static void main(String[] args) {
        final Samsung galaxy = new Samsung(imeiNumber: "1020394959697", modelNumber: "M31", screenSize: "6.1", batteryCapacity: "3600", cameraMegapixel: "12");

        final Iphone xs = new Iphone(imeiNumber: "1202101201202", modelNumber: "XS", screenSize: "5.9", batteryCapacity: "3000", iosVersion: "16.1.2");

        BigInteger phoneNumber = new BigInteger(val: "9822877994");
        String email = "roshish152002@gmail.com";

        galaxy.voice_call(phoneNumber);
        galaxy.video_call(email);

        xs.voice_call(email);
        xs.video_call(phoneNumber);
    }
}
```

Output:

```
Calling the phone number: 9822877994
Video calling the user with email: roshish152002@gmail.com
Calling the user with email: roshish152002@gmail.com
FaceTiming the phone number: 9822877994
```

This code defines three classes: Mobile, Samsung, and iPhone. The Mobile class represents a mobile phone with some basic characteristics like its IMEI number, model number, screen size, and battery capacity. It also has two methods called "voice_call" and "video_call" that can be used to make phone calls or video calls to a phone number or email address. The Samsung and iPhone classes are subclasses of the Mobile class and they represent specific brands of mobile phones. The Samsung class has an additional field called "cameraMegapixel" that represents the megapixel of the camera on the phone. The Iphone class has an additional field called "iosVersion" that represents the version of the iOS operating system on the phone. The Iphone class also overrides the "video_call" method from the Mobile class to provide its own implementation of the method that prints a different message when making a video call.

The Consumer class has a main method that creates instances of the Samsung and Iphone classes and calls the "voice_call" and "video_call" methods on them.

3. How can you validate your college email address and the password [**Herald@#-787%College12]**
    a. Explain in code.

Note: Above password ends with numeric value, starts with capital letter, can contain capital letter in middle and some special characters.

```java
package Week4_Workshop.Validate_Regex;

import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Validation {
    static boolean validateEmail(String email) {
        String regex = "^np+[0-9]+cs+[0-9]+s+[0-9]{6}+@heraldcollege\\.edu\\.np$";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(email);
        return matcher.matches();
    }

    static boolean validatePassword(String password) {
        String regex = "^[A-Z][A-Za-z0-9@$!%*?&]*[0-9]$";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(password);
        return matcher.matches();
    }
}

class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String email, password;

        while (true) {
            System.out.println(x: "Enter your email: ");
            email = scanner.nextLine();
            if (Validation.validateEmail(email)) {
                break;
            }
            System.out.println(x: "Invalid Email! Please Try Again!");
        }

        while (true) {
            System.out.println(x: "Enter your password: ");
            password = scanner.nextLine();
            if (Validation.validatePassword(password)) {
                break;
            }
            System.out.println(x: "Invalid Password! Please Try Again!");

        }

        System.out.println(Validation.validatePassword(password) ? "Valid Password" : "Invalid Password");
        scanner.close();
    }
}
```

Output:

```
Enter your email:
roshish152002@gmail.com
Invalid Email! Please Try Again!
Enter your email:
np03cs4s220181@heraldcollege.edu.np
Enter your password:
herald@123
Invalid Password! Please Try Again!
Enter your password:
Herald@123
Valid Password
```

This code defines two classes: Validation and Main. The Validation class has two static methods called "validateEmail" and "validatePassword" that are used to check if a given email address or password are valid or not.

The "validateEmail" method takes an email address as a parameter and checks if it matches a specific regular expression. In this case, the regular expression defines the pattern that a valid email address should follow. If the email address matches the pattern, the method returns "true", otherwise it returns "false".

The "validatePassword" method takes a password as a parameter and checks if it matches a specific regular expression. The regular expression defines the pattern that a valid password should follow. If the password matches the pattern, the method returns "true", otherwise it returns "false".

The Main class has a main method that prompts the user to enter their email and password. It then uses the "validateEmail" and "validatePassword" methods from the Validation class to check if the entered values are valid. If the entered values are not valid, it prompts the user to try again. If the entered values are valid, it prints a message saying that the password is valid.

4. How can you handle exceptions? Explain from code.

Note: You have to explain the code you have done.

```java
package Week4_Workshop.Exception_Handling;

import java.util.InputMismatchException;
import java.util.Scanner;

class InputHandler {
    private String name;
    private int age;

    public InputHandler(String name, int age) {
        this.name = name;
        this.age = age;
    }

    static InputHandler takeInput() {
        String name;
        int age = -1;

        Scanner scanner = new Scanner(System.in);
        System.out.println(x: "Enter your name: ");
        name = scanner.nextLine();
        System.out.println(x: "Enter your age: ");

        try {
            age = scanner.nextInt();
        } catch (InputMismatchException e) {
            System.out.println(e);
        }
        scanner.close();
        return new InputHandler(name, age);
    }

    public void displayResult() {
        System.out.println(name);
        if (age == -1) {
            System.out.println(x: "You entered invalid age!");
        } else {
            System.out.println(age);
        }
    }
}

class Main {
    Run | Debug
    public static void main(String[] args) {

        InputHandler input = InputHandler.takeInput();
        input.displayResult();

    }
}
```

Output:

```
Enter your name:
Roshish
Enter your age:
Invalid
java.util.InputMismatchException
Roshish
You entered invalid age!
```

```
Enter your name:
Roshish
Enter your age:
20
Roshish
20
```

This code defines two classes: InputHandler and Main. The InputHandler class has two fields: "name" and "age", which represent the name and age of a person. It also has a static method called "takeInput" that prompts the user to enter their name and age. It uses a Scanner object to read the input from the user and stores it in the "name" and "age" fields.

The "takeInput" method also has a try-catch block that catches any "InputMismatchException" that might be thrown while trying to read the age from the user. An "InputMismatchException" is thrown when the input provided by the user cannot be matched to the expected data type. In this case, if the user enters a non-integer value for their age, the exception is caught and a message is printed to the console.

The InputHandler class also has a method called "displayResult" that displays the name and age of the person. If the age is -1, it means that an invalid value was entered by the user and a message is printed to the console saying "You entered invalid age!".

The Main class has a main method that creates an instance of the InputHandler class and calls the "takeInput" and "displayResult" methods on it. This causes the program to prompt the user for their name and age and display the entered values on the console.