Create a class Animal with properties like name and age. Create a subclass Dog that inherits from Animal and adds a property breed. Demonstrate the use of constructors in both the Animal and Dog classes

```
~

class Animal {
    String name;
    int age;
    Animal(String name) {
        this.name = name;
    }
}

class Dog extends Animal {
    String breed;
    Dog(String name, int age, String breed) {
        super(name);
        this.breed = breed;
        this.age = age
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog("somedog", 3, "some species");
    }
}
```

Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle

```
~

class Shape {
    double getPerimeter() {
        return 0;
    }

    double getArea() {
        return 0;
    }
}

class Circle extends Shape {
    double radius;
    Circle(double radius) {
        this.radius = radius;
    }

    @Override
    double getPerimeter() {
        return 2 * 3.141592653 * radius;
    }

    @Override
    double getArea() {
        return 3.141592653 * radius * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
    }
}
```

Extend the Animal and Dog example by adding a constructor to the Animal class that takes a name parameter. In the Dog class, use the super keyword to call the constructor of the Animal class. Create instances of Dog and demonstrate the use of the super keyword.

```
 ~

class Animal {
    String name;
    int age;
    Animal(String name) {
        this.name = name;
    }
}

class Dog extends Animal {
    String breed;
    Dog(String name, int age, String breed) {
        super(name);
        this.breed = breed;
        this.age = age;
    }
}

public class main {
    public static void main(String[] args) {
        Dog dog = new Dog("somedog", 3, "some species");
        System.out.println(dog.age);
    }
}
```

Create a class Person with a protected attribute address. Extend it with a subclass Employee that adds a department attribute. Demonstrate how the protected keyword allows access to the address property in the Employee subclass.

```
 ~

class Person{
    protected String adress;
}

class Employee extends Person {
    String department;

}
public class main {
    public static void main(String[] args) {
        Employee emp = new Employee();
        System.out.println(emp.adress);
    }
}
```

Create class Parent with a private variable, a protected variable, and a public variable. Create a subclass Child and demonstrate how each type of variable is accessed (or not accessed) within the subclass

```
 ~


class Parent{
    private String a;
    protected String b;
    public String c;
}

class Child extends Parent{
}
public class main {
    public static void main(String[] args) {
        Child emp = new Child();
        System.out.println(emp.a);
        System.out.println(emp.b);
        System.out.println(emp.c);
    }
}
```

```
 ~
[wizard@archlinux 3]$ javac main.java
main.java:12: error: a has private access in Parent
        System.out.println(emp.a);
                           ^
1 error
```

Create a final class FinalClass. Attempt to extend it with another class and observe the compiler error. Also, create a final method within a class and try to override it in a subclass.

```
 ~

final class Parent{
    private String a;
    protected String b;
    public String c;
}

class Child extends Parent{
    final private void something() {
        System.out.println("to override");
    }
}

class another extends Child {
    @override
    private void something(){
        System.out.println("to override");
    }
}
public class main {
    public static void main(String[] args) {
        Child emp = new Child();
        another emp1 = new another();

    }
}
```

```
 ~

[wizard@archlinux 3]$ javac main.java
main.java:7: error: cannot inherit from final Parent
class Child extends Parent{
                    ^
main.java:14: error: cannot find symbol
    @override
     ^
  symbol:   class override
  location: class another
2 errors
[wizard@archlinux 3]$
```

Create a class ,'Calculator' that should be able to perform addition operations for both integers and doubles. Implement the following steps:

- Create an instance of the Calculator class.
- Use the add method to add two integers (e.g., 5 and 8) and display the result.
- Call the add method with three integers (e.g., 10, 15, and 20) and display the result.
- Use the add method to add two doubles (e.g., 3.5 and 2.7) and display the result.
- Call the add method with three doubles (e.g., 1.1, 2.2, and 3.3) and display the result.

```
 ~

class Calculator {
    public void add(int a, int b) {
        System.out.println(a+b);
    }

    public void add(double a, double b) {
        System.out.println(a+b);
    }

    public void add(int a, int b,int c) {
        System.out.println(a+b+c);
    }

    public void add(double a, double b,double c) {
        System.out.println(a+b+c);
    }
}

public class main {
    public static void main(String[] args) {
        Calculator clacc = new Calculator();
        clacc.add(1,2);
        clacc.add(1,2,3);
        clacc.add(1.1,2.2);
        clacc.add(1.1,2.2,3.3);

    }
}
```

```
 ~

[wizard@archlinux 3]$ java main
3
6
3.3000000000000003
6.6
[wizard@archlinux 3]$
```

Section 2:

The signup process must not be successful until:

- The full name is enter. The length must be greater than four.
- The mobile number has 10 digits starting with 0.
- The Password must initiate with capital alphabets followed by either one of @/& and ending with numeric.
- The password confirmation matches the initial entered password.
- The DOB is in the format DD/MM/YYYY.
- The user is at least 21 years old. The age should be calculated based on the year entered in the DOB (Only consider year)

~

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class User {
    private String fname;
    private String number;
    private String password;
    private String dob;

    public void setField(String field, String value) {
        switch (field) {
            case "fname" -> this.fname = value;
            case "number" -> this.number = value;
            case "password" -> this.password = value;
            case "dob" -> this.dob = value;
        }
    }

    @Override
    public String toString() {
        return String.format( "User's name: %s, User's number: %s, User's password: %s, User's birth date: %s", this.fname, this.number, this.password, this.dob);
    }
}

class Signup {
    public static boolean validateField(String fieldName, String value) {
        return switch (fieldName) {
            case "fname" -> value.length() > 4;
            case "number" -> value.matches("0\\d{9}");
            case "password" -> value.matches("[A-Z].*[@&].*\\d$");
            case "dob" -> {
                if (!value.matches("\\d{2}/\\d{2}/\\d{4}")) {
                    yield false;
                }
                int birthYear = Integer.parseInt(value.split("/")[2]);
                yield (2024 - birthYear) ⩾ 21;
            }
            default -> false;
        };
    }
}

public class main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<User> allUsers = new ArrayList<>();
        String[] fields = {"fname", "number", "password", "dob"};

        while (true) {
            User user = new User();
            for (String field : fields) {
                while (true) {
                    System.out.print(String.format("Enter %s: ",field));
                    String input = scanner.nextLine();

                    if (field.equals("password")) {
                        System.out.print("Confirm password: ");
                        String confirmPassword = scanner.nextLine();
```

```
                if (!input.equals(confirmPassword)) {
                    System.out.println("Passwords do not match. Try again.");
                    continue;
                }
            }

            if (Signup.validateField(field, input)) {
                user.setField(field, input);
                break;
            } else {
                System.out.println("Invalid " + field + ". Please try again.");
            }
        }
    }

    allUsers.add(user);
    System.out.println("User created: " + user);

    System.out.print("Sign Up Another? y/n: ");
    String choice = scanner.nextLine();

    if (!choice.equalsIgnoreCase("y")) {
        break;
    }
}

System.out.println("All Users:");
for (User u : allUsers) {
    System.out.println(u);
}

scanner.close();
    }
}
```

```
 ~
[wizard@archlinux 3]$ java main
Enter fname: swo
Invalid fname. Please try again.
Enter fname: someone
Enter number: 0111111111
Enter password: Asom@123
Confirm password: Asome@123
Passwords do not match. Try again.
Enter password: Asom@123
Confirm password: Asom@123
Enter dob: 22/10/2000
User created: User's name: someone, User's number: 0111111111, User's password: Asom@123, User's
birth date: 22/10/2000
Sign Up Another? y/n: n
All Users:
User's name: someone, User's number: 0111111111, User's password: Asom@123, User's birth date:
22/10/2000
```