



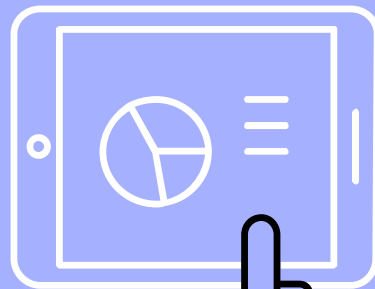
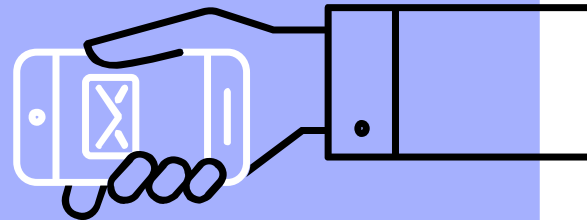
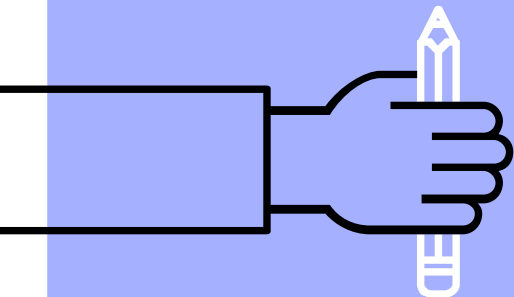
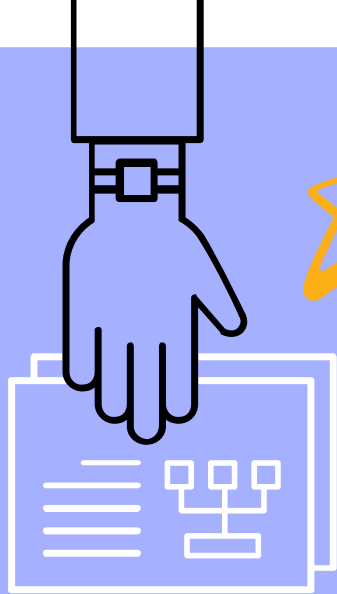
Universidade

Cruzeiro do Sul

Programação Orientada a Objetos

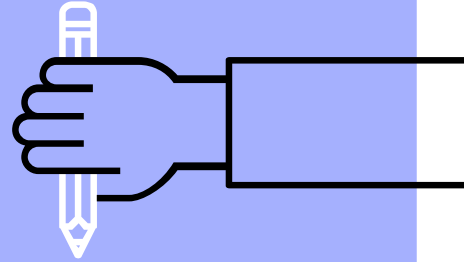
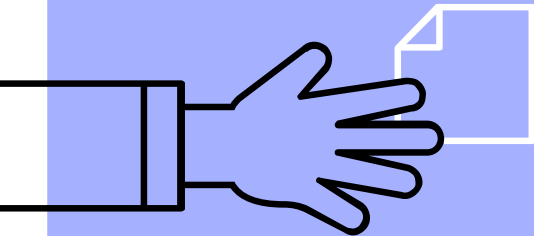
Aula 09

Prof.^a Erika Miranda



9. Unidade

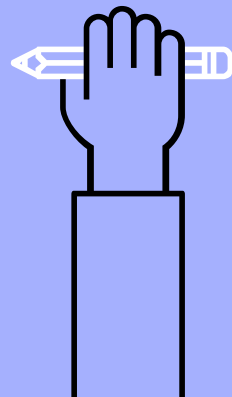
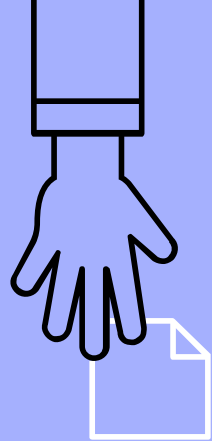
Interface



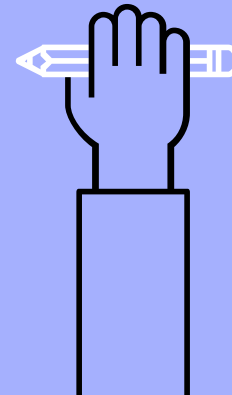
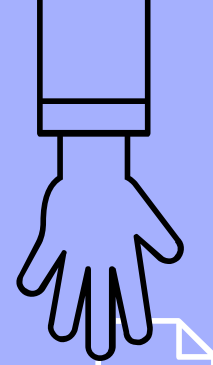
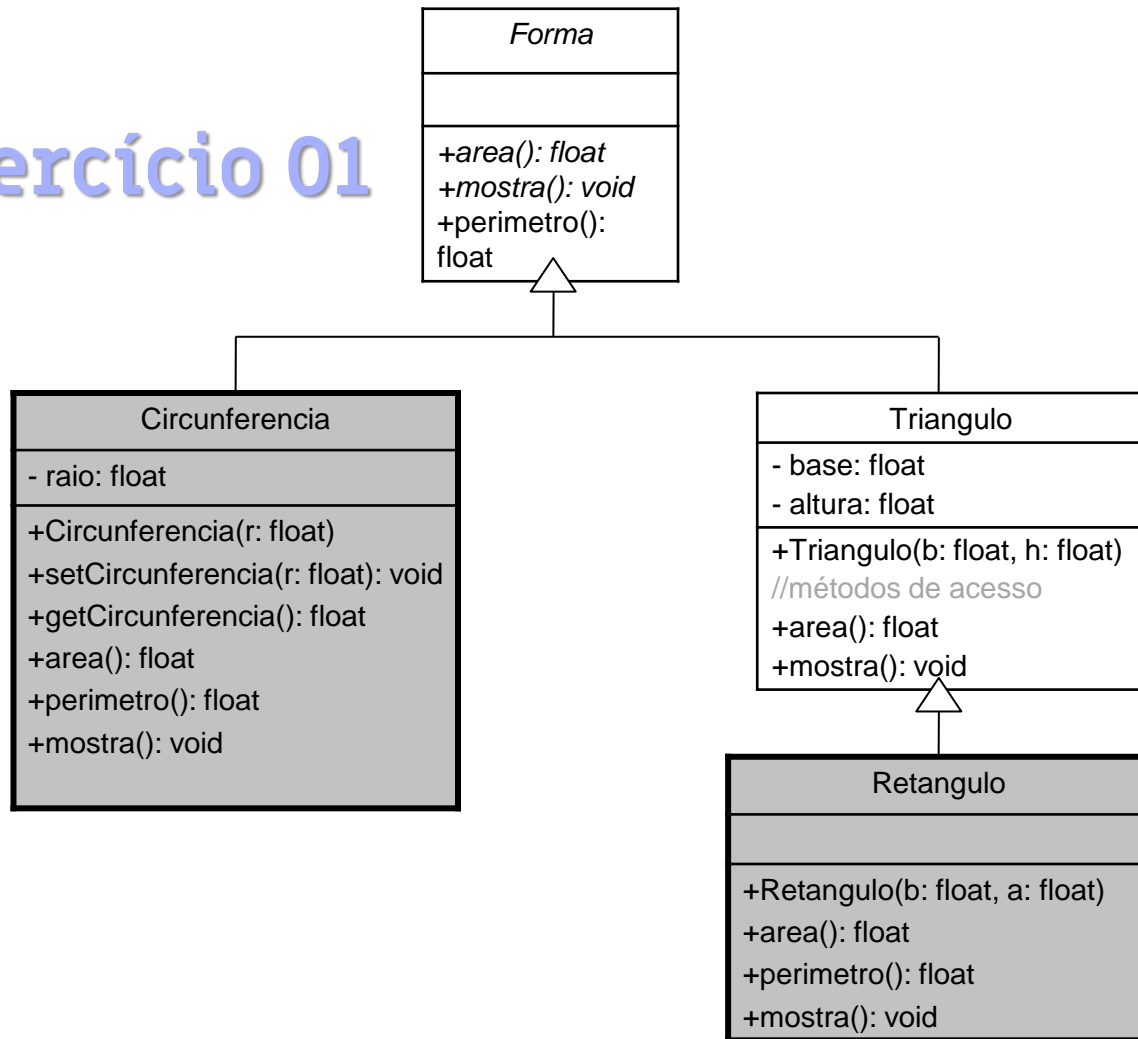
Interfaces x Classes Abstratas

Use classes abstratas quando você quiser definir um *"template"* para subclasses e você possui alguma implementação (métodos concretos) que todas as subclasses podem utilizar.

Use interfaces quando você quiser definir uma regra que todas as classes que implementem a interface devem seguir, independentemente se pertencem a alguma hierarquia de classes ou não.



Exercício 01



Exercício 01

Crie a classe abaixo como subclasse de Forma:

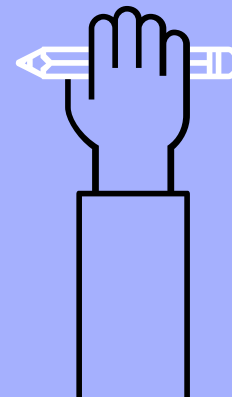
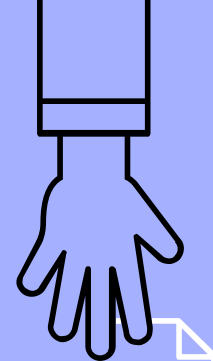
Circunferencia
- raio: float
+Circunferencia(r: float) +setCircunferencia(r: float): void +getCircunferencia(): float +area(): float +perimetro(): float +mostra(): void

O método `area()` deve retornar valor da área da circunferência, sabendo que $area = \pi * r^2$

O método `perimetro()` deve retornar o valor do perímetro: $perimetro = 2 * \pi * r$

Em ambos os métodos utilize a constante `Math.PI` da classe `Math`.

O método `mostra` deve exibir os valores de todos os atributos da classe



Exercício 01

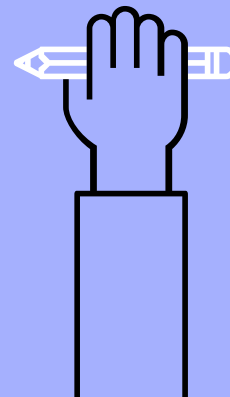
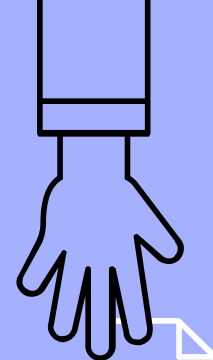
Crie a classe abaixo como subclasse de Triangulo:

Retangulo
+Retangulo(b: float, a: float)
+area(): float
+perimetro(): float
+mostra(): void

O método `area()` deve retornar valor da área da circunferência, sabendo que $\text{area} = \text{base} * \text{altura}$

O método `perimetro()` deve retornar o valor do perímetro: $\text{perimetro} = (\text{base} * \text{altura}) * 2$

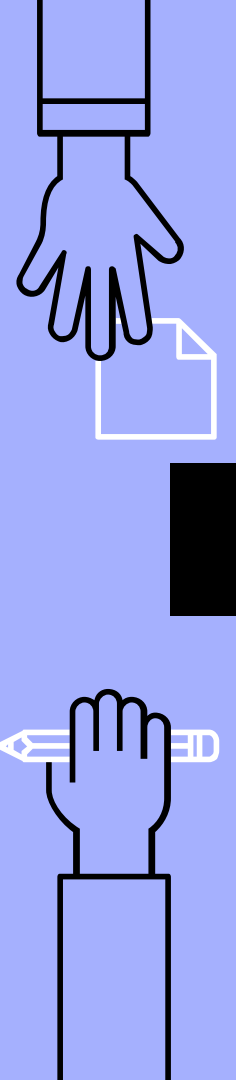
O método `mostra` deve exibir os valores de todos os atributos da classe



Exercício 01

Instancie dois objetos na classe java principal, um da classe Circunferencia e outro da classe Retangulo, com os valores dos atributos digitados pelo usuário e utilize o construtor com parâmetros.

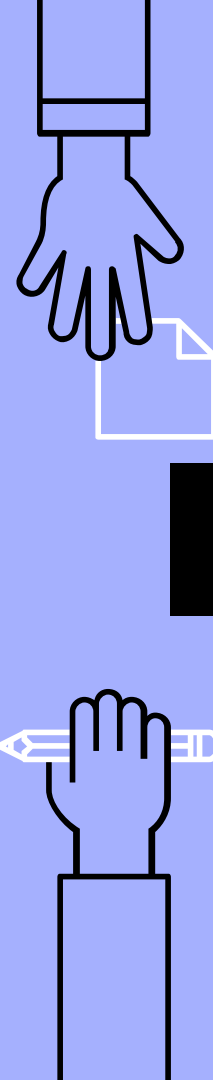
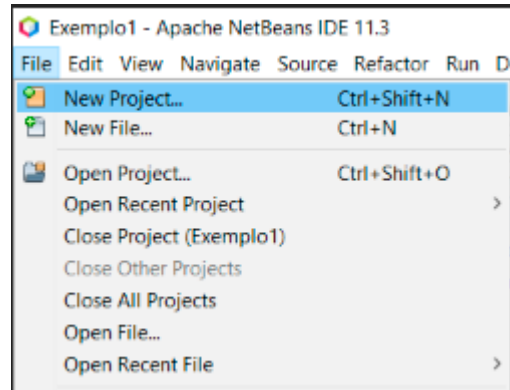
Mostre os dados de cada objeto através do método mostra().



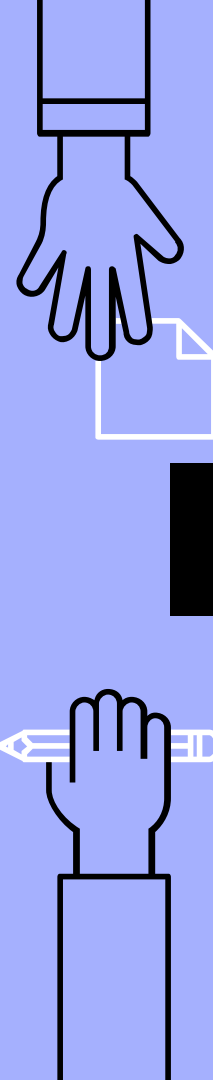
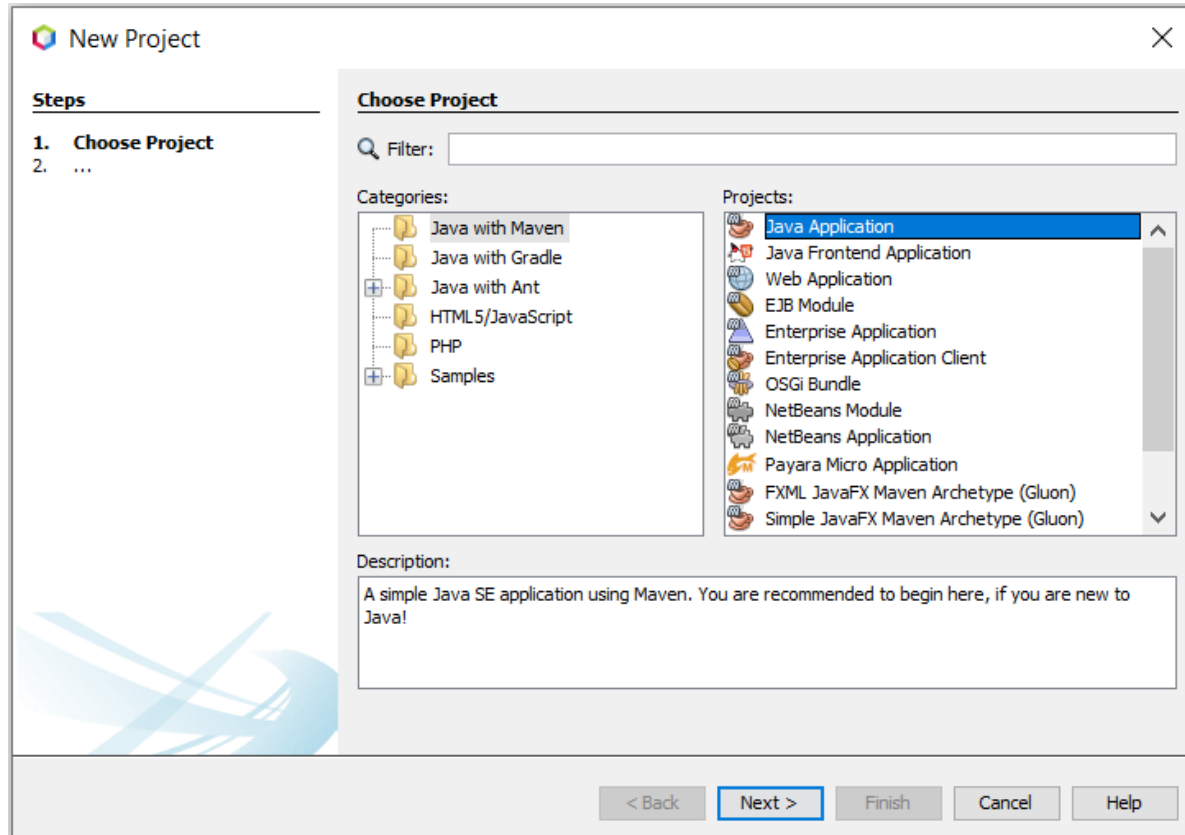
Exercício 01

Vamos implementar usando o NetBeans.

Vamos iniciar um novo projeto:

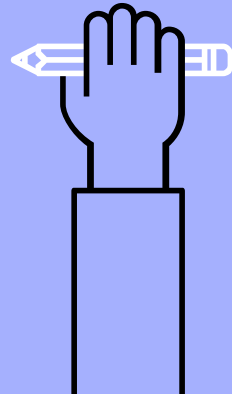
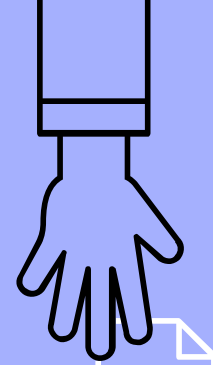


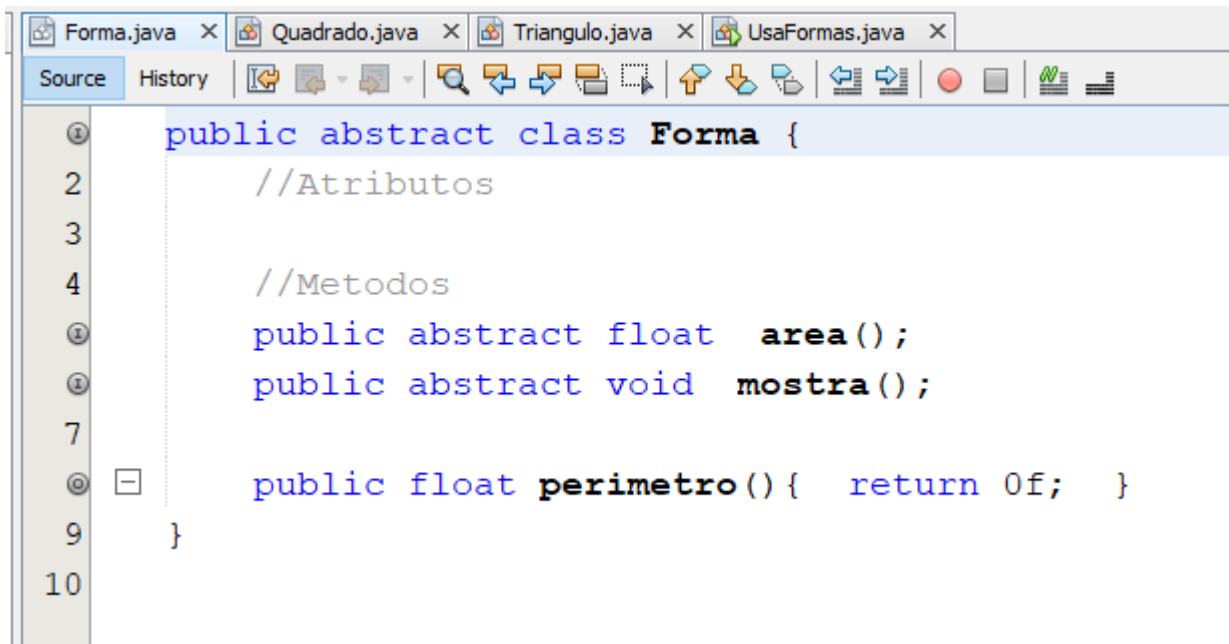
Exercício 01



Vamos criar as seguintes Classes:

- ▶ **Forma**(Classe Abstrata)
- ▶ **Quadrado**(extends Forma)
- ▶ **Triangulo**(extends Forma)
- ▶ **UsaFormas**(public static void main(String[] args))

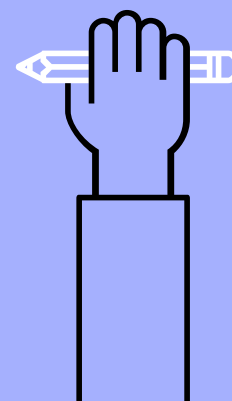
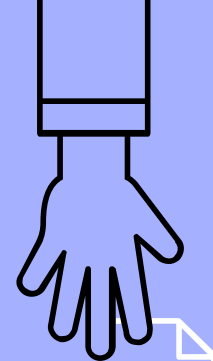




The image shows a screenshot of a Java IDE with four open files: Forma.java, Quadrado.java, Triangulo.java, and UsaFormas.java. The 'Forma.java' file is active, displaying the following code:

```
1 public abstract class Forma {  
2     //Atributos  
3  
4     //Metodos  
5     public abstract float area();  
6     public abstract void mostra();  
7  
8     public float perimetro(){ return 0f; }  
9 }  
10
```

The code defines an abstract class **Forma** with two abstract methods, **area()** and **mostra()**, and one concrete method, **perimetro()**, which returns 0f. The IDE interface includes a toolbar with various icons for file operations and a sidebar with a file explorer.



Forma.java x Quadrado.java x Triangulo.java x UsaFormas.java x

Source History

```
1 public class Quadrado extends Forma {
2     //Atributos
3     private float base;
4
5     //Construtor
6     public Quadrado(float b){ base = b; }
7
8     //Metodos de acesso
9     public float getBase() { return base; }
10    public void setBase(float b) { base = b; }
11
12    //sobreposição do método da classe Pessoa
13    public float perimetro() {
14        return base * 4;
15    }
16
17    //Implementação dos métodos abstratos da classe Forma
18    public float area() {
19        return base * base;
20    }
21    public void mostra() {
22        System.out.println("Base: " + base + "\nPerimetro: " + perimetro() + "\nArea: " + area());
23    }
24 }
```

```
1 public class Triangulo extends Forma {
2     //Atributos
3     private float base, altura;
4
5     //Construtor
6     public Triangulo(float b, float h){
7         base = b;
8         altura = h;
9     }
10
11     //Metodos de acesso
12     public float getBase() { return base; }
13     public float getAltura() { return altura; }
14     public void setBase(float b) { base = b; }
15     public void setAltura(float h) { altura = h; }
16
17     //Implementação dos métodos abstratos da classe Forma
18     public float area() {
19         return (base * altura)/2;
20     }
21     public void mostra() {
22         System.out.println("\nBase: " + base + "\nAltura: " + altura + "\nArea: " + area());
23     }
24 }
```

```
Forma.java x Quadrado.java x Triangulo.java x UsaFormas.java x
Source History
1 import java.util.*;
2 public class UsaFormas{
3     public static void main(String args[]){
4         float b,a;
5         Quadrado q;
6         Triangulo t;
7
8         Scanner scan = new Scanner(System.in);
9
10        System.out.print("Digite a base do quadrado: ");
11        b = scan.nextFloat(); //para String use o nextLine()
12        q = new Quadrado(b);
13        //na chamada do metodo abaixo e passado um objeto da classe Quadrado
14        q.mostra();
15
16        System.out.print("Digite a base do triangulo: ");
17        b = scan.nextFloat();
18        System.out.print("Digite a altura do triangulo: ");
19        a = scan.nextFloat();
20        t = new Triangulo(b,a);
21        //na chamada do metodo abaixo e passado um objeto da classe Triangulo
22        t.mostra();
23    }
24 }
```

