Compartilhar o seu link com: luciorocha @ professores.utfpr.edu.br

Séfora< Aula 15 - POO >
mabylly< Cópia de Aula 15 - POCO4A - Exercícios propostos >
Angélica < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Filipe Augusto Parreira Almeida < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Matheus HIrata < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Rafael Kendy Naramoto Lopes < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Roberto Furlani Neto < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Raphael Uematsu < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Marcos Tadao Shoji< Cópia de Aula 15 - POCO4A - Exercícios propostos >
Vitor Luiz de Castro Viana Cópia de Aula 15 - POCO4A - Exercícios propostos
Guilherme Ramalho
<https://docs.google.com/document/d/14t35GaihwRvjarmTnj4U5znFKqj6WzEZENFoOw1YDi0/edit>
Henrique Cois < Aula15(20/04)POO >
Plinio < Cópia de Aula 15 - POCO4A - Exercícios propostos >
Gabriel Takeshi < Cópia de Aula 15 - POCO4A - Exercícios propostos >
João Pedro Cavani Meireles< Cópia de Aula 15 - POCO4A - Exercícios propostos >
Rodrigo Leandro Benedito Cópia de Aula 15 - POCO4A - Exercícios propostos
Julio Farias< Cópia de Aula 15 - POCO4A - Exercícios propostos >
João Pedro de Paula:< Cópia de Aula 15 - POCO4A - Exercícios propostos >
Felipe F B Lorusso < Aula 15 - POCO4A >
Felipe Antonio Magro: Cópia de Aula 15 - POCO4A - Exercícios propostos

```
//Implementação dos slides de sala


public class Principal {

   public abstract class Animal {

      public abstract void caminhar();
   }

   public abstract class Anfibio extends Animal {

      protected String escamas;

   }

   public abstract class Terrestre extends Animal {
```

```
    protected String pelos;

}

public class Leao extends Terrestre {

    private String movimento;

    public Leao(String pelos) {
        super();   //construtor da superclasse
        this.pelos = pelos;
        this.movimento = "ANDAR";
    }

    public void rugir() {
        System.out.println("RRR!");
    }

    public void caminhar() {
        System.out.println(this.movimento);
    }

    public String toString() {
        return "LEAO " + this.movimento;
    }
}

public class Sapo extends Anfibio {

    private String movimento;

    public Sapo(String escamas) {
        super();   //construtor da superclasse
        this.escamas = escamas;
        this.movimento = "PULAR";
    }

    public void coachar() {
        System.out.println("Coach!");
    }

    public void caminhar() {
        System.out.println(this.movimento);
    }

    public String toString() {
        return "SAPO " + this.movimento;
```

```
    }
  }

  public Principal() {

    Animal animal = new Sapo("POUCAS"); //Subclasse eh um objeto da Superclasse
    animal.caminhar();     //PULAR
    ((Sapo) animal).coachar();   //

    animal = new Leao("MUITOS"); //Subclasse eh um objeto da Superclasse
    animal.caminhar();   //ANDAR
    ((Leao) animal).rugir();

    animal = new Sapo("MUITAS");
    animal.caminhar();  //PULAR

    Sapo sapo = new Sapo("POUCAS");
    //sapo = animal;    //ERRO.     Superclasse ⟶ Subclasse
    sapo = ((Sapo) animal);


    ArrayList<Animal>  lista = new ArrayList<>();
    lista.add( new Sapo("POUCAS"));
    lista.add( new Leao("MUITOS"));

    for( Animal animal : lista )
        animal.caminhar();


  }

  public static void main(String[] args) {
    Principal principal = new Principal();

  }

}
```

//Exemplo 1: Software de cadastro de bicicletas. O sistema cadastra bicicletas do tipo montain bike, Speed, ergonômica.

Nomes (classes): Bicicleta, MontainBike, Speed, Ergonomica
Atributos (variáveis de instância): cor
Verbos (métodos): cadastrar

```
import java.util.ArrayList;

public class Principal {
public abstract class Bicicleta {

        public abstract void cadastrar();
}

public class MontainBike extends Bicicleta {
      private String cor;
      public MontainBike(String cor){
         super(); //construtor padrao
         this.cor = cor;
      }
      public void cadastrar(){   System.out.println("MONTAIN BIKE"); }

}
public class Speed extends Bicicleta {
      private String cor;
      public Speed(String cor){
           super(); //construtor padrao
           this.cor = cor;
      }
      public void cadastrar(){   System.out.println("SPEED"); }

}
public class Ergonomica extends Bicicleta {
      private String cor;
      public Ergonomica(String cor){
            super(); //construtor padrao
            this.cor = cor;
      }
      public void cadastrar(){   System.out.println("ERGONOMICA"); }
}
```

```
        public Principal(){
                ArrayList<Bicicleta> lista = new ArrayList<>();
                Bicicleta   bicicleta  = new MontainBike("AZUL"); //Superclasse ←Subclasse
                lista.add( bicicleta  );

                bicicleta = new Speed("AMARELA");
                lista.add( bicicleta );

                bicicleta = new Ergonomica("VERDE");
                lista.add( bicicleta );

                for( Bicicleta b : lista )    b.cadastrar();
        }//fim construtor
        public static void main( String [ ] args ){
                new Principal();
        }
}
```

//Exemplo 2: Relação entre subclasse e superclasse.

```
Animal   animal  =  new Sapo();     //Superclasse ← Subclasse.
                                    // Sapo eh um Animal

Sapo  sapo   = animal;     // ERRO. Animal NAO EH um Sapo.
                                    //Pode ser um Leao, Coelho, Sapo…

Bicicleta  bicicleta = new MontainBike();    //Superclasse ← Subclasse
                                             //MontainBike EH UMA Bicicleta

MontainBike  mb =  bicicleta;  // ERRO.
                                  //Pode ser Speed, Ergonomica, MontainBike…

MontainBike  mb =  ( (MontainBike) bicicleta);

//Resumo: SEMPRE eh possivel atribuir um objeto da subclasse para um objeto da
superclasse.
//              NEM SEMPRE eh possivel atribuir um objeto da superclasse para um objeto da
subclasse ( faltam campos).
```

//Classe Concreta
//Exemplo 3: Sistema de cadastro de Bicicletas com Classe Concreta. (Lista de Tipo Concreto)

```
import java.util.ArrayList;

public class Principal {
public class Bicicleta {

        public void cadastrar() {  System.out.println(“GENERICA”); }
}

public class MontainBike extends Bicicleta {
      private String cor;
      public MontainBike(String cor){
          super(); //construtor padrao
          this.cor = cor;
      }
      @Override
      public void cadastrar(){   System.out.println("MONTAIN BIKE"); }

}
public class Speed extends Bicicleta {
      private String cor;
      public Speed(String cor){
            super(); //construtor padrao
            this.cor = cor;
      }
      @Override
      public void cadastrar(){   System.out.println("SPEED"); }

}
public class Ergonomica extends Bicicleta {
      private String cor;
      public Ergonomica(String cor){
             super(); //construtor padrao
             this.cor = cor;
      }
      @Override
      public void cadastrar(){   System.out.println("ERGONOMICA"); }
}
       public Principal(){
```

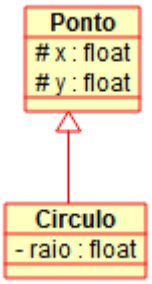
```
            ArrayList<Bicicleta> lista = new ArrayList<>();
            Bicicleta   bicicleta  = new MontainBike("AZUL"); //Superclasse ←Subclasse
            lista.add( bicicleta  );

            bicicleta = new Speed("AMARELA");
            lista.add( bicicleta );

            bicicleta = new Ergonomica("VERDE");
            lista.add( bicicleta );

            for( Bicicleta b : lista )    b.cadastrar();
        }//fim construtor
        public static void main( String [ ] args ){
                new Principal();
        }
}
```

//Polimorfismo com Interfaces
//Exemplo 4: Sistema de cadastro de Bicicletas. (Lista de Tipo Abstrato)

```
import java.util.ArrayList;

public class Principal {
public interface Bicicleta {

        public abstract void cadastrar();
}

public class MontainBike implements Bicicleta {
      private String cor;
      public MontainBike(String cor){
          super(); //construtor padrao
          this.cor = cor;
      }
      @Override
      public void cadastrar(){   System.out.println("MONTAIN BIKE"); }

}
public class Speed implements Bicicleta {
      private String cor;
      public Speed(String cor){
```

```
            super(); //construtor padrao
            this.cor = cor;
      }
      @Override
      public void cadastrar(){   System.out.println("SPEED"); }

}
public class Ergonomica implements Bicicleta {
      private String cor;
      public Ergonomica(String cor){
            super(); //construtor padrao
            this.cor = cor;
      }
      @Override
      public void cadastrar(){   System.out.println("ERGONOMICA"); }
}
       public Principal(){
              ArrayList<Bicicleta> lista = new ArrayList<>();
              Bicicleta   bicicleta  = new MontainBike("AZUL"); //Superclasse ←Subclasse
              lista.add( bicicleta  );

             bicicleta = new Speed("AMARELA");
             lista.add( bicicleta );

             bicicleta = new Ergonomica("VERDE");
             lista.add( bicicleta );

             for( Bicicleta b : lista )    b.cadastrar();
        }//fim construtor
        public static void main( String [ ] args ){
               new Principal();
        }
}
```

//Exemplo 5: Formas Geométricas



//Circulo eh um Ponto

```
public class Principal {

//Superclasse
public class Ponto {
    protected float x;
    protected float y;

    public Ponto(float x, float y){
        this.x = x;
        this.y = y;
    }
}

//Subclasse
public class Circulo extends Ponto {
      private float raio;
      public Circulo(float x, float y, float raio ){
            super( x, y );
            this.raio = raio;
     }
}
//Subclasse
public class Linha extends Ponto {
      private float x1, x2;
      private float y1, y2;
      public Linha(float x, float y ){
            super( x, y );
            this.x1 = x;
            this.x2 = x+1;
            this.y1 = y;
            this.y2 = y+1;
     }
}
```

```
        public Principal(){
              ArrayList<Ponto> lista = new ArrayList<>();
              Ponto  ponto = new Circulo(1f,2f,3f);   //Circulo eh um Ponto
              lista.add (ponto);

              Circulo circulo = new Circulo(1f,2f,3f);
              //circulo = ponto;     //ERRO. Ponto NAO EH um Circulo

              circulo = ((Circulo) ponto);    //OK
              lista.add( circulo );

              ponto = new Linha(1f, 2f);  //Polimorfismo. Linha EH UM Ponto
              lista.add( ponto );


        }
        public static void main(String [ ] arg s ){

                new Principal();
        }
}
```
