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

Link da Pasta Google Drive da Disciplina: POCO4A - 2s2022

Notas da Lista 1: POCO4A - 2s2022 - Notas Parciais

Aluno1:
Alexandre Calisto: Aula 10 - 13/09/2022
Iago Macarini: POO - Aula 10
Michael Pariz Pereira: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Fernando Rafael: POCO4A - Aula 10 - 13/09/2022 - Exercícios
Vitor Luis de Queiroz Batista: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
RAFAEL ZAUPA WATANABE: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Matheus Henrique de A. Pereira: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Guilherme Conceição Ramalho: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Gabriel Candelária: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Camila Costa: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Raphael Hideyuki Uematsu: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Felipe Antonio Magro: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Rodrigo Leandro Benedito: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Arnald Souza: Aula 10 - 13/09/2022
Gabriel Abe/Plinio Koiama: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Leonardo G. Fagote: Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Ruan Perondi Urbanjos: Copy of POCO4A - Aula 10 - 13/09/2022 - Exercícios
Lucas Santana: Aula10 - Lucas Santana
Gustavo Nunes : Cópia de POCO4A - Aula 10 - 13/09/2022 - Exercícios
Gustavo Naoki Jodai Kurozawa: POCO4A - Aula 10
Ruan Mateus Trizotti: Aula 10
Maria Eduarda Pedroso: Pedroso - POCO4A - Aula 10 - 13/09/2022 - Exercícios

---

Exemplo1:

Interface: é um tipo abstrato de dados. Ela define quais métodos a classe que a implementa deve utilizar. -A interface não implementa os métodos.

Classe Abstrata: Generaliza os métodos das subclasses. A classe abstrata não deve ser implementada. A classe abstrata “pode” definir a implementação de interfaces. Caso a classe abstrata não implemente o método da interface, ela se torna abstrata. A classe abstrata não

precisa ter todos os métodos abstratos. Pelo menos 1 (um) método deve ser declarado abstrato.

herança de métodos abstratos.

```
public interface IAnimal {

    public abstract void andar();
    public int getPernas();
    public bool temPelos();
    public bool temOlhos();
}
public interface ICoelho {
    public abstract void andar();
    public int getPernas();
    public bool temPelos();
}
public abstract class Animal implements IAnimal {
        protected final String tipo = “MAMIFERO”;

        //@Override
        public final bool temPelos(){
            return true;
        }
       public final bool temOlhos(){
            return true;
       }
}




public final class Coelho extends Animal implements ICoelho {

    public void andar(){
          System.out.println(“ANDAR COELHO”);
    }
    public int getPernas(){
          return 4;
    }
  /*   public bool temPelos(){
           return true;
    }
  */
}
```

```
public final class Sapo implements IAnimal {
    public void andar(){
       System.out.println("ANDAR SAPO");
    }
    public getPernas(){
        return 2;
    }
    //Sobrescrita
    public bool temPelos(){
       return false;
    }
}

public class Principal {

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

      public void iniciar(){
             Coelho  coelho = new Coelho();
             coelho.andar();
             System.out.println( coelho.temPelos() );
      }

}
```

Exercícios propostos:

1) Observe a Figura 1 a seguir:

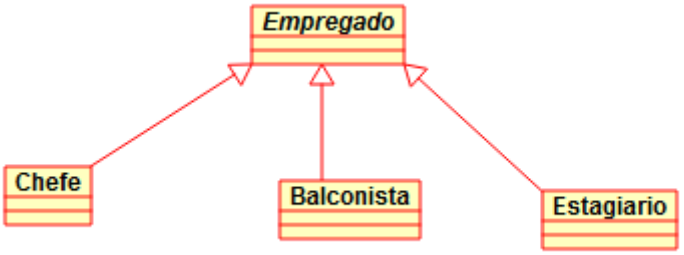


Figura 1: Diagrama UML de Classes.

Classes: Empregado, Chefe, Balconista, Estagiario

a) Inclua no projeto a classe Data.

b) Modifique o código para incluir a variável de instância 'private Data dataNascimento' na classe Empregado. Inclua um método acessor e um mutador para essa nova variável de instância.

c) Não devem ser criados novos métodos nas classes Chefe, Balconista e Estagiario, porém, modifique o construtor de cada uma dessas Classes para incluir a Data de nascimento do funcionário.

d) Adicione apenas métodos acessores na Classe Data para cada uma das variáveis de instância.

e) Suponha que a folha de pagamento seja processada uma vez por mês. Crie um vetor de objetos Empregado para armazenar referências a vários objetos de funcionários.

f) Crie a interface IEmpregado que será implementada pela Classe Balconista.

```
public interface IEmpregado {
    public abstract void setPagamento();
    public abstract String toString();
}

public abstract class Empregado implements IEmpregado {
    protected float salario;
    private Data dataNascimento;

    public Empregado( Data  dataNascimento ){
        setDataNascimento( dataNascimento );
    }
    public Data getDataNascimento(){
        return this.dataNascimento;
    }
    public void setDataNascimento(Data  dataNascimento ){
        this.dataNascimento = dataNascimento;
```

```
    }
     public String toString(){   return this.getClass().getSimpleName() +
                                          " Salario: " + this.salario +
                                         "\nData de nascimento: " + this.dataNascimento );
}
public final class Chefe extends Empregado {

     public Chefe(Data dataNascimento){
        super( dataNascimento );

        setPagamento();
      }
     public void setPagamento(){
          this.salario = 1000;
     }

}
public final class Balconista extends Empregado {

     public Balconista(Data dataNascimento){
        super( dataNascimento );

        setPagamento();
      }

     public void setPagamento(){
          this.salario = 100;
     }

}
public final class Estagiario extends Empregado {

     public Estagiario(Data dataNascimento){
        super( dataNascimento );

        setPagamento();
      }
     public void setPagamento(){
          this.salario = 10;
     }

}

public final class Data {
    private int dia;
    private int mes;
    private int ano;
```

```
    public Data(int dia, int mes, int ano){
            setDia( dia );
            setMes( mes );
            setAno( ano );
    }
    public int getDia(){  return this.dia; }
    public int getMes(){  return this.mes; }
    public int getAno(){  return this.ano; }

    private void setDia( int dia ){  this.dia = dia; }
    private void setMes( int mes ){  this.mes = mes; }
    private void setAno( int ano ){  this.ano = ano; }

    public String toString(){
         return this.dia + "/" + this.mes + "/" + this.ano;
    }
}

public class Principal {

    public Principal(){
        Empregado [ ] lista = new Empregado[5];  //Vetor é estático
        lista[0] = new Balconista(  new Data(13, 9, 2022) );
        lista[1] = new Chefe( new Data( 12, 9, 2022) );
        lista[2] = new Estagiario( new Data( 14, 9, 2022) );

        for(int i=0; i<lista.length; i++)
            System.out.println( lista[ i ] );

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

}
```

2) Faça a implementação Orientada a Objetos do problema anunciado a seguir:

   a) Crie 3 (três) classes não relacionadas por herança: Construcao, Carro e Bicicleta.

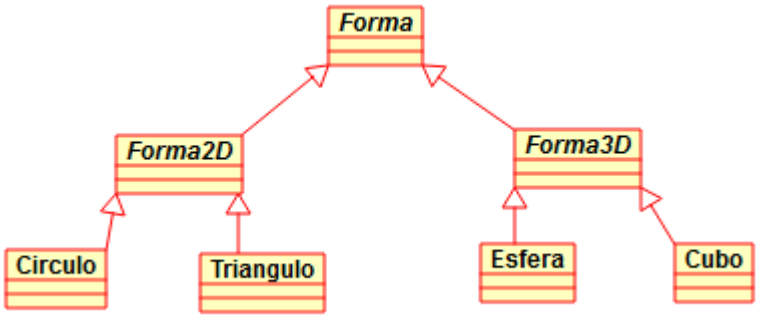
   b) Dê a cada Classe atributos e comportamentos únicos que não estão presentes em outras classes.

c) Crie a Interface EmissaoCarbono com um método getEmissaoCarbono.

d) Cada Classe deve implementar a Interface EmissaoCarbono.

e) Invoque o método getEmissaoCarbono de cada objeto.

3) Observe a Figura 2 a seguir:



a) Implemente a hierarquia de Classes mostrada na Figura. Apenas as Classes folha são Classes concretas, as demais são classes abstratas.

b) A Classe Forma2D deve conter o método getArea.

c) A Classe Forma3D dever conter os métodos getArea e getVolume.

d) Crie uma Classe Principal que tenha um vetor de Formas com objetos de cada Classe concreta.

e) O programa deve imprimir o tipo de cada objeto instanciado.