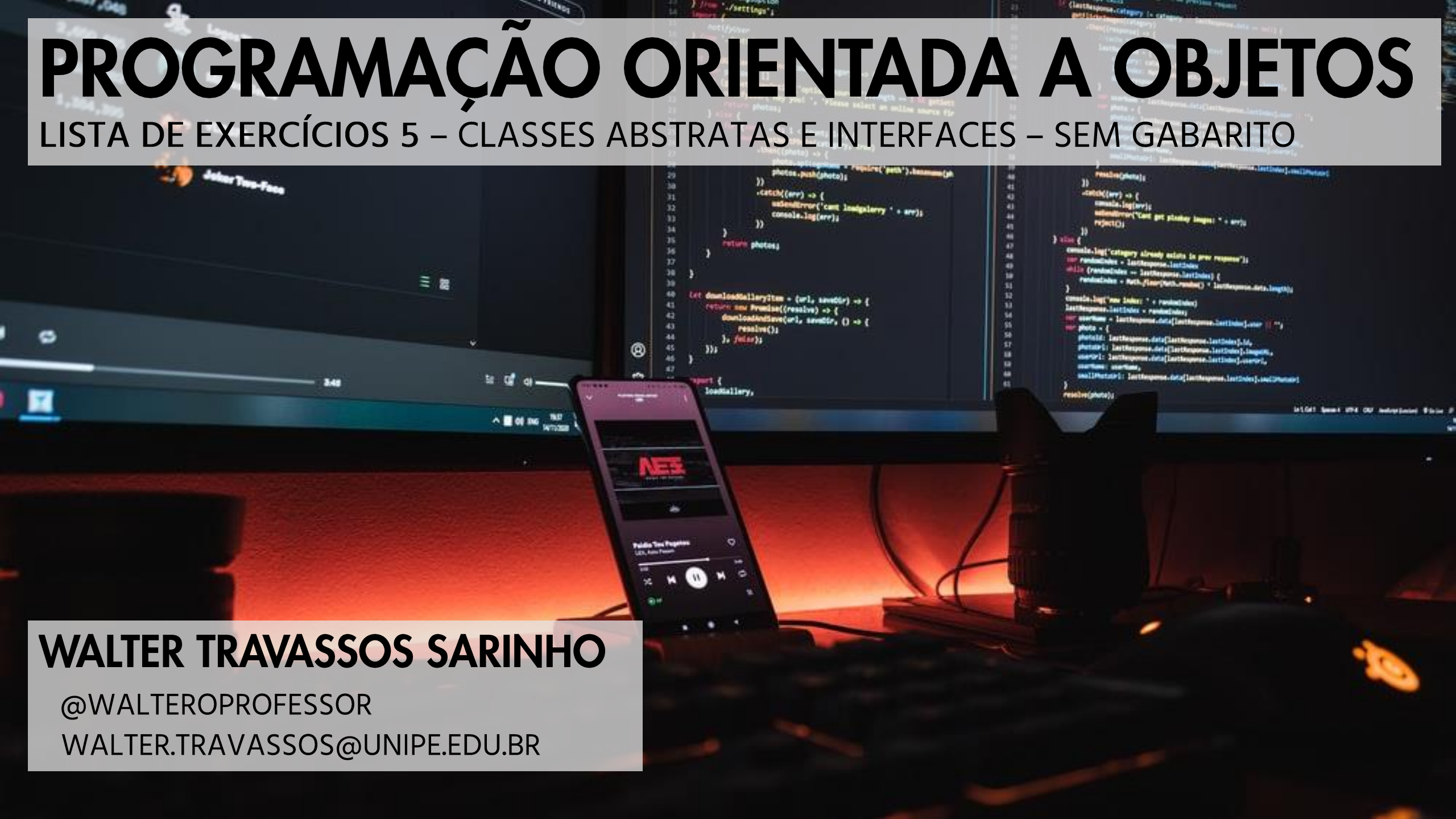
PROGRAMAÇÃO ORIENTADA A OBJETOS

LISTA DE EXERCÍCIOS 5 – CLASSES ABSTRATAS E INTERFACES – SEM GABARITO



WALTER TRAVASSOS SARINHO

@WALTEROPROFESSOR

WALTER.TRAVASSOS@UNIPE.EDU.BR

Lista de Exercícios 5

Rey

Classes Abstratas e Interfaces

Exercício 1

Durante o desenvolvimento de uma aplicação orientada a objetos com Java, um Técnico criou uma interface para obrigar um conjunto de classes de diferentes origens a implementar certos métodos de maneiras diferentes, viabilizando a obtenção de polimorfismo. A interface criada pelo Técnico pode

a) Conter apenas métodos implementados.
b) Ser instanciada diretamente.
c) Possuir um único construtor vazio.
d) Possuir assinatura de métodos públicos.
e) Conter variáveis e métodos privados.

Exercício 2

Em Java as interfaces são tipos especiais de classes que podem conter

a) Atributos e métodos setter e getter implementados.
b) Métodos privados e protegidos.
c) Somente métodos estáticos.
d) Assinaturas de métodos.
e) Diversos construtores.

Exercício 3

- Uma classe do tipo interface possui apenas declaração de métodos estáticos públicos sem aplicação concreta, razão por que só pode ser implementada em classes abstratas.

- Certo ou Errado?

Exercício 4

Considerando a versão 1.7 do JDK da linguagem de programação Java, classes nesta linguagem podem ser estendidas. Sobre esse assunto, assinale a alternativa INCORRETA.

a) Ao criar um objeto de uma subclasse, o primeiro passo é executar o corpo do construtor da subclasse.

b) A extensão de classes pode ser usada para diversos objetivos. É mais comumente usada para especialização.

c) Sempre que estende-se uma classe, cria-se uma nova classe com um contrato expandido. Entretanto, o contrato herdado não sofre alteração.

d) Quando um objeto é criado, é alocada memória para todos os seus campos, incluindo aqueles herdados de superclasses e cujos campos são configurados com o valor default de seus respectivos tipos.

e) Nesta linguagem, uma subclasse pode estender de apenas uma superclasse.

Exercício 5

A programação orientada a objetos é baseada em diversos conceitos, tais como encapsulamento, herança, polimorfismo e abstração. Com relação a esses conceitos, é correto afirmar que:

a) O conceito de encapsulamento é alcançado por meio da definição da visibilidade pública aos atributos e métodos.

b) Herança é um mecanismo que permite que uma classe herde todo o comportamento e os atributos de outra classe. Em Java, pode-se implementar tanto a herança única quanto a herança múltipla

c) Interface pode ser considerada como a forma com que um objeto se apresenta para outros, no que diz respeito aos seus atributos e métodos. Em Java, uma mesma classe não pode implementar mais de uma interface.

d) Polimorfismo é o uso de um mesmo nome para identificar diferentes implementações dos métodos. Seu uso é comum na definição de construtores, em que os mesmos podem ser implementados em diferentes versões para as diferentes formas de se instanciar a classe.

e) Para uma classe ser considerada abstrata, todos os seus métodos devem ser abstratos. Em Java, para se definir uma classe abstrata deve-se utilizar a palavra chave abstract no início de sua declaração.

Exercício 6

A assinatura do método desenhar() na classe abstrata permite uma implementação diferente nas classes que herdam a classe Poligono. A sobrescrita deste método nas classes filhas refere-se ao conceito de:

a) encapsulamento.
b) especialização.
c) herança.
d) agregação.
e) polimorfismo.

```
abstract public class Poligono {
    abstract public void desenhar ();
}
public class Quadrado extends Poligono {
    public void desenhar () {...}
}
public class Tetraedro extends Poligono {
    public void desenhar () {...}
}
Poligono [100] v;
v [32] = new Quadrado ();
v [56] = new Tetraedro ();
v [32].desenhar ();
```

Exercício 7

As linhas que contêm a instrução s = f.calcularSalario(); demonstram um conceito da orientação a objetos conhecido como:

a) encapsulamento.
b) sobrecarga de métodos.
c) polimorfismo.
d) sobrescrita de construtores.
e) métodos abstratos.

```
public class Funcionario {
private int id;
private String nome;
private double valorBase;
    public Funcionario() {
    }
    public Funcionario(int id, String nome, double valorBase) {
        this.id = id;
        this.nome = nome;
        this.valorBase=valorBase;
    }
    public double getValorBase() {
        return valorBase;
    }
    public double calcularSalario(){
        return valorBase;
    }
}
public class Mensalista extends Funcionario{
private double descontos;
    public Mensalista(double descontos, int id, String nome, double
    valorBase) {
        super(id, nome, valorBase);
        this.descontos = descontos;
    }
@Override
    public double calcularSalario(){
        return super.getValorBase() - descontos;
    }
}
public class Diarista extends Funcionario {
    private int diasPorSemana;
    public Diarista( int diasPorSemana, int id, String nome, double
    valorBase) {
        super(id, nome, valorBase);
        this.diasPorSemana = diasPorSemana;
    }
    @Override
    public double calcularSalario(){
        return super.getValorBase() * diasPorSemana;
    }
}
```

Em uma classe principal foram digitadas, no interior do método `main`, as seguintes linhas:

```
double s;
Funcionario f;
f=new Diarista(3,10456,"Ana Maria",90);
s = f.calcularSalario();
System.out.println(s);
f=new Mensalista(298.56,10457,"Pedro Henrique",877.56);
s = f.calcularSalario();
System.out.println(s);
```

Exercício 8

Um dos conceitos mais importantes da orientação a objetos é o de interface. Interfaces podem reduzir o acoplamento entre as classes e tornar o código mais reutilizável. Sobre as interfaces em Java, analise as alternativas e marque a correta.

a) É possível implementar um método usando o modificar default, no entanto, é mais comum implementá-los apenas nas classes que implementam a interface.

b) Podem ser implementadas apenas por classes que utilizam em sua declaração a palavra extends.

c) Não podem ter campos de instância e também não permitem a especificação de constantes.

d) Podem ser instanciadas de outras classes, desde que estejam no mesmo pacote.

e) Podem ter seus objetos convertidos no tipo classe diretamente, sem a realização de typecasting.

Exercício 9

Analise a classe a seguir retirada de uma aplicação Java que contém as classes Empregado.java, Diarista.java e Start.java. A classe Start.java possui o método main:

```
package geral;
 public class Diarista extends Empregado{
   private int diasTrabalhados;
   public Diarista(int diasTrabalhados, int id, String nome, double valorReferencia) {
       super(id, nome, valorReferencia);
       this.diasTrabalhados = diasTrabalhados;
}
 public int getDiasTrabalhados() {
       return diasTrabalhados;
}
public void setDiasTrabalhados(int diasTrabalhados) {
       this.diasTrabalhados = diasTrabalhados;
}
@Override
 public double obterSalario(){
       return super.getValorReferencia() * diasTrabalhados;
}
 public double obterSalario(double percentualAcrescimo) {
       double salario= this.obterSalario();
       double salarioReajustado = salario + salario * percentualAcrescimo /100;
       return salarioReajustado;
}
 public double obterSalario(double adicional, double desconto){
       return this.obterSalario() + adicional - desconto;
}
}
```

Exercício 9 - continuação

Analisando o código do slide anterior, é possível concluir que:

a) Essa classe está incorreta, pois não é possível criar diversos métodos com o mesmo nome, como é o caso do método obterSalario.

b) O método obterSalario() foi sobrescrito da classe Empregado

c) A instrução super(id, nome, valorReferencia); passa os valores contidos nas variáveis locais id, nome e valorReferencia para o método chamado super, da classe Empregado.

d) A instrução super.getValorReferencia() chama o método getValorReferencia() da classe Start.java, a classe principal que inicializa a aplicação.

e) A existência de vários métodos na classe Diarista com o nome obterSalario caracteriza um conceito conhecido como sobrescrita de métodos.

Exercício 10

O que é impresso como resultado da execução do programa composto pelas classes acima?

a) | X| Z|| X||| X| Z|| Y||
b) | X|||| X||| X|||| X||
c) (| X| Z|| X||(| X| Z|| Y||
d) | X| Z|| X||| X| Z|| Z||
e) | X| Z|| X||| X| Z|| X||

```
public class X {
	public void op1() {
		System.out.print("|X|Z");
		op2();
	}

	public void op2() {
		System.out.print("||X||");
	}

	public static void main(String args[]) {
		X xxx = new X();
		xxx.op1();

		X yyy = new Y();
		yyy.op1();
	}
}
```

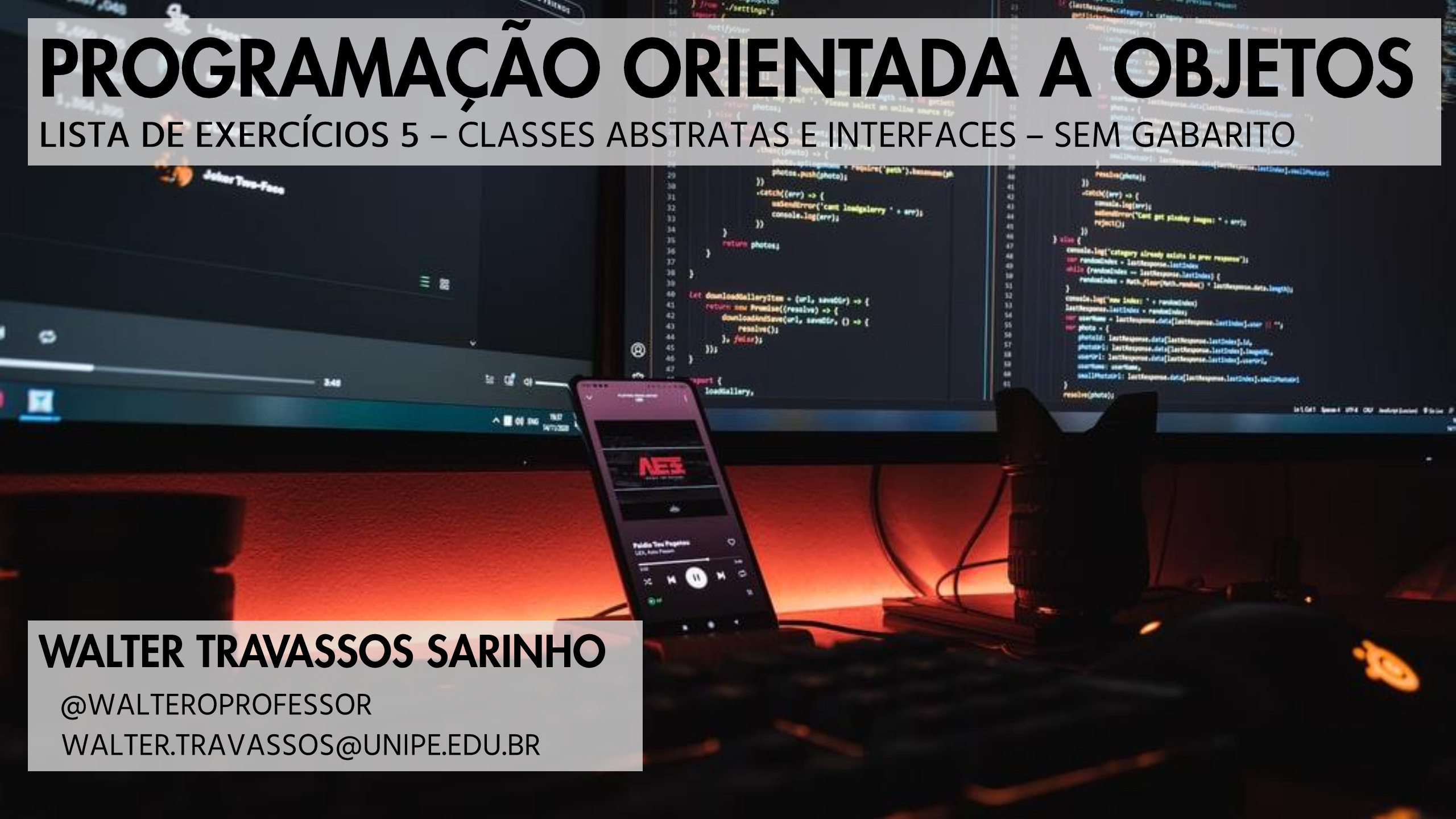
```
public class Y extends X {
	public void op2() {
		System.out.print("||Y||");
	}

	public void op2(int p) {
		System.out.print("||" + p + "||");
	}

}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

LISTA DE EXERCÍCIOS 5 – CLASSES ABSTRATAS E INTERFACES – SEM GABARITO



WALTER TRAVASSOS SARINHO

@WALTEROPROFESSOR

WALTER.TRAVASSOS@UNIPE.EDU.BR