UNIVERSIDADE da MADEIRA

RESOLUÇÃO DE EXAME ANTERIOR III

Mónica Cameirão
monica.cameirao@staff.uma.pt

Grupo I - 1

Pretende-se implementar uma aplicação para apoio à gestão de uma companhia de teatro, a qual tem vários atores. Todos os atores são caraterizados pelo **nome** e **categoria** (que varia de 1 a 4), dados que têm de fornecer ao entrar na companhia de teatro; além disso os atores são também caraterizados por um **número** único que a aplicação atribui automaticamente de forma sequencial quando os atores entram na companhia. Os atores podem ser principais ou secundários. Os atores secundários são caraterizados pelo número de **papéis realizados**; os atores principais são caraterizados por **prémios de desempenho** monetários. Todos os atores têm um método **calculaSalario** que calcula e devolve o valor do salário de um ator. No entanto, dependendo do tipo de ator (principal ou secundário), o salário é recalculado para contemplar os extras inerentes a cada tipo de ator. A companhia de teatro deverá disponibilizar um método **totalSalarios** que calcula e devolve o montante total em salários a pagar aos seus atores. Esta deverá ter também um método **afiliar** que afilia novos atores à companhia de teatro.

1. Elabore o diagrama de classes UML para o problema apresentando. Não é necessário incluir no diagrama eventuais métodos seletores e modificadores. Indique o nome das relações que utilizou e o significado de cada uma delas.

Grupo I - 2

2. Defina em JAVA a classe que representa o ator. Inclua os métodos que considerar apropriados. Implemente o método **calculaSalario**; o salário é calculado multiplicando a categoria por um **valor fixo** que é constante e tem o valor de 1000 euros. Imprima o número e o nome de um ator na mesma linha.

```
public class Ator {
    //Variáveis de instância
    private String nome;
    private int categoria;
    private static int seguinte; // variável estática para atribuir nr sequencial
    private int numero;

    private final double VALOR_FIXO = 1000;

    // Construtor
    public Ator(String nome, int categoria){
        this.nome = nome;
        this.categoria = categoria;
        seguinte ++;  //incrementa quando é criado um novo ator
        numero = seguinte;
    }
```

continua

Grupo I - 2

2. Defina em JAVA a classe que representa o ator. Inclua os métodos que considerar apropriados. Implemente o método **calculaSalario**; o salário é calculado multiplicando a categoria por um **valor fixo** que é constante e tem o valor de 1000 euros. Imprima o número e o nome de um ator na mesma linha.

```
    // Cálculo do salário
    public double calculaSalario(){
        return VALOR_FIXO*categoria;
    }

    //Método toString
    public String toString(){
        return "Nr: " + numero + " Nome: " + nome;
    }
}
```

Grupo I - 3

3. Defina a classe que representa o ator secundário. Inclua os métodos que considerar apropriados. O salário de um ator secundário é calculado somando ao salário de um ator 20% desse salário por cada papel realizado. Além do número e do nome do ator, imprima também o valor do salário.

```
public class AtorSecundario extends Ator {
    // Variáveis de instância
    private int papeisRealizados;

    //Construtor
    public AtorSecundario(String nome, int categoria, int papeisRealizados){
        super(nome,categoria);
        this.papeisRealizados = papeisRealizados;
    }

    //Cálculo do salario
    public double calculaSalario(){
        return super.calculaSalario() + super.calculaSalario()*0.2*papeisRealizados;
    }

    //Método toString
    public String toString(){
        return super.toString() + " Salario: " + calculaSalario() +"\n";
    }
}
```

Grupo I - 4

4. Defina a classe que representa o ator principal. Inclua os métodos que considerar apropriados. O salário de um ator principal é calculado somando ao salário de um ator o valor referente ao prémio de desempenho. Além do número e do nome do ator, imprima também o valor do salário.

```
public class AtorPrincipal extends Ator {
    // Variáveis de instância
    private double premio;

    //Construtor
    public AtorPrincipal(String nome, int categoria, double premio){
        super(nome,categoria);
        this.premio = premio;
    }

    //Cálculo do salario
    public double calculaSalario(){
        return super.calculaSalario() + premio;
    }

    //Método toString
    public String toString(){
        return super.toString() + " Salario: " + calculaSalario()+"\n";
    }
}
```

Grupo I - 5

5. Defina a classe que representa a companhia de teatro. Implemente o método **totalSalarios** e o método **afiliar**. Deverá imprimir a informação de todos os atores que fazem parte da companhia de teatro.

```
import java.util.ArrayList;

public class CompanhiaTeatro {
    // Variáveis de instância
    private ArrayList<Ator> atores;

    // Construtor
    public CompanhiaTeatro(){
        atores = new ArrayList();
    }

    // Método que calcula o total dos salário
    public double totalSalarios(){
        double total=0;
        if (atores.size()>0){
           for (Ator a : atores)
               total += a.calculaSalario();
        }
        return total;
    }
```

continua

Grupo I - 5

5. Defina a classe que representa a companhia de teatro. Implemente o método **totalSalarios** e o método **afiliar**. Deverá imprimir a informação de todos os atores que fazem parte da companhia de teatro.

```
    //Método para afiliar um novo ator
    public void afiliar(Ator ator){
        atores.add(ator);
    }

    // Método toString
    public String toString(){
        return "Atores: \n" + atores;
    }
}
```

Grupo I - 6

6. Defina a classe principal onde é criada uma companhia de teatro, na qual são afiliados um ator principal e um ator secundário. Use uma referência polimórfica. A aplicação deverá mostrar no ecrã o salário total a pagar pela companhia de teatro aos seus atores.

```
public class ProjetoTeatro {

    public static void main(String[] args) {

        CompanhiaTeatro cp = new CompanhiaTeatro();

        Ator a;
        a = new AtorPrincipal("Marta",4,5000);
        cp.afiliar(a);

        a = new AtorSecundario("Paulo",2,2);
        cp.afiliar(a);
        System.out.println(cp);

        System.out.println("A companhia pagou "+cp.totalSalarios()+" em salários");
    }
}
```

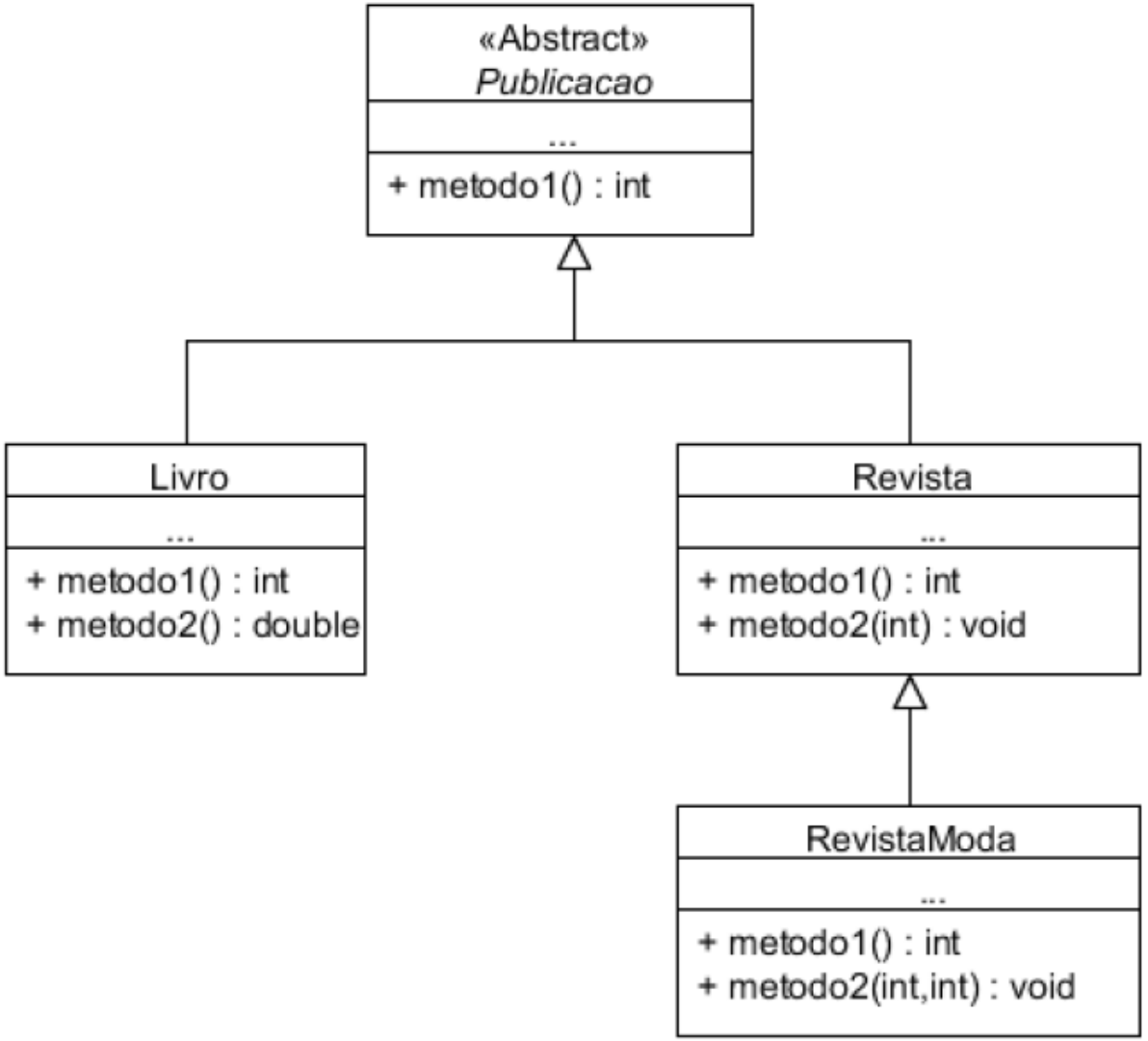
Grupo II

1. De que duas formas podemos utilizar o polimorfismo em Java? Descreva cada um dos casos e dê exemplos.

2. Qual a diferença entre variáveis de instância e variáveis estáticas? E entre métodos de instância e métodos estáticos? Discuta o âmbito de aplicação de cada caso.

3. O que é um tipo genérico? Descreva e dê um exemplo.

Grupo III

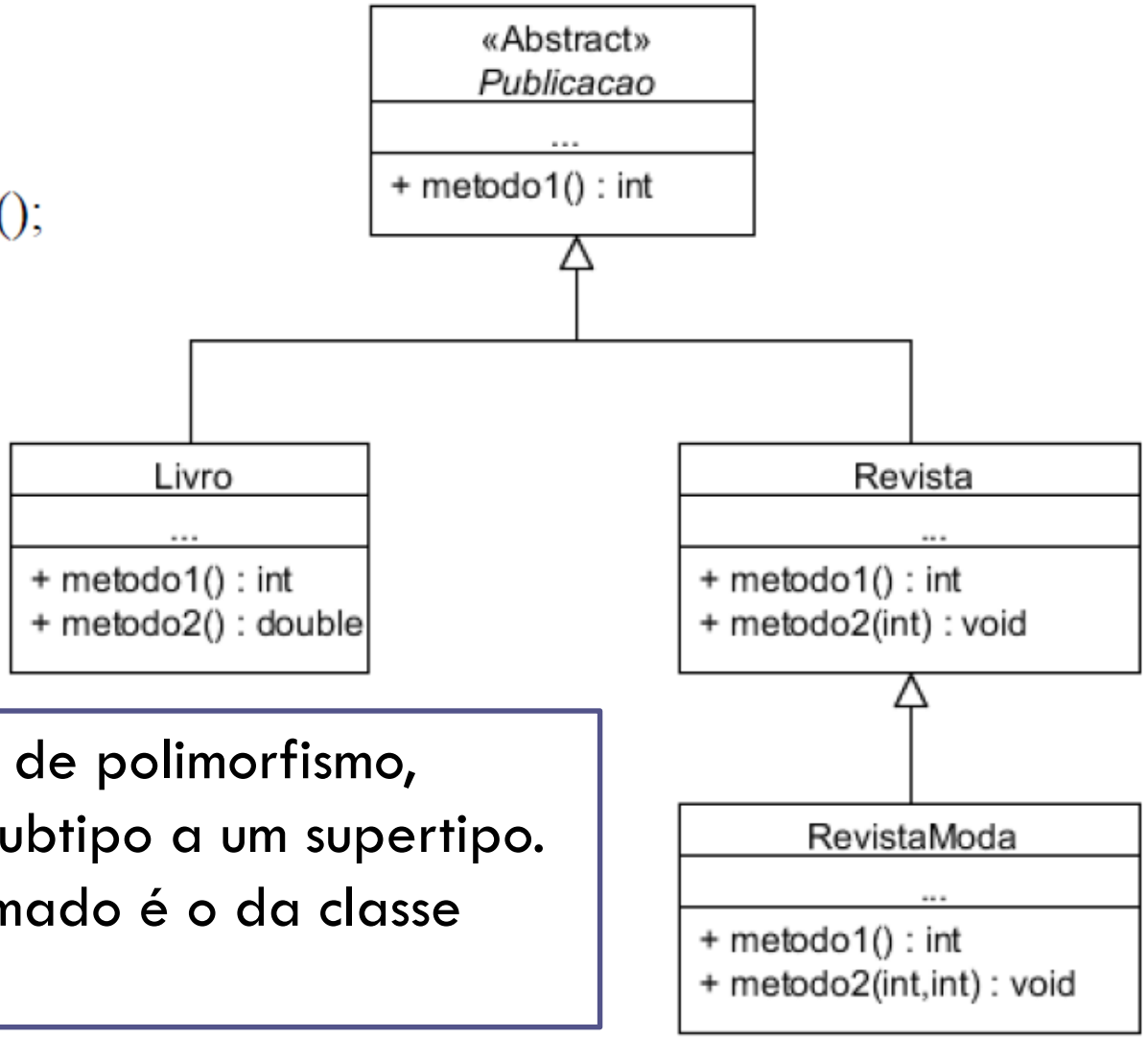


1. Indique em que situações se verifica *overriding* e *overloading* de métodos. Explique porquê.

Grupo III

2. Para cada um dos blocos seguintes indique se o mesmo é ou não válido. Deverá justificar cada caso.

a.
```
Publicacao p;
Livro d = new Livro();
p = d;
p.metodo1();
```

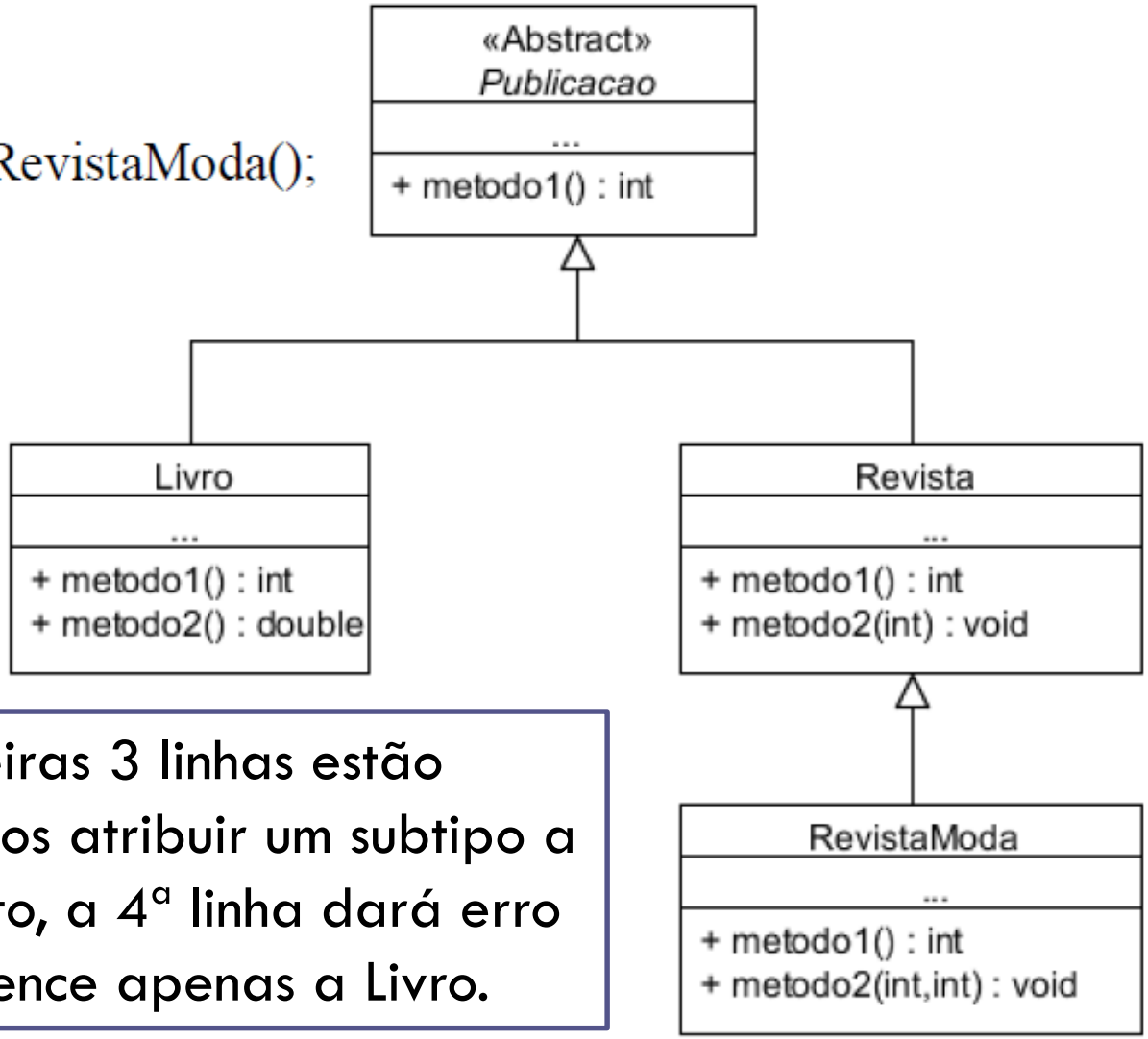


VÁLIDO. Pelo conceito de polimorfismo, podemos atribuir um subtipo a um supertipo. O metodo1 a ser chamado é o da classe Livro.

Grupo III

2. Para cada um dos blocos seguintes indique se o mesmo é ou não válido. Deverá justificar cada caso.

b.
```
Publicacao p;
RevistaModa rm = new RevistaModa();
p = rm;
p.metodo2();
```

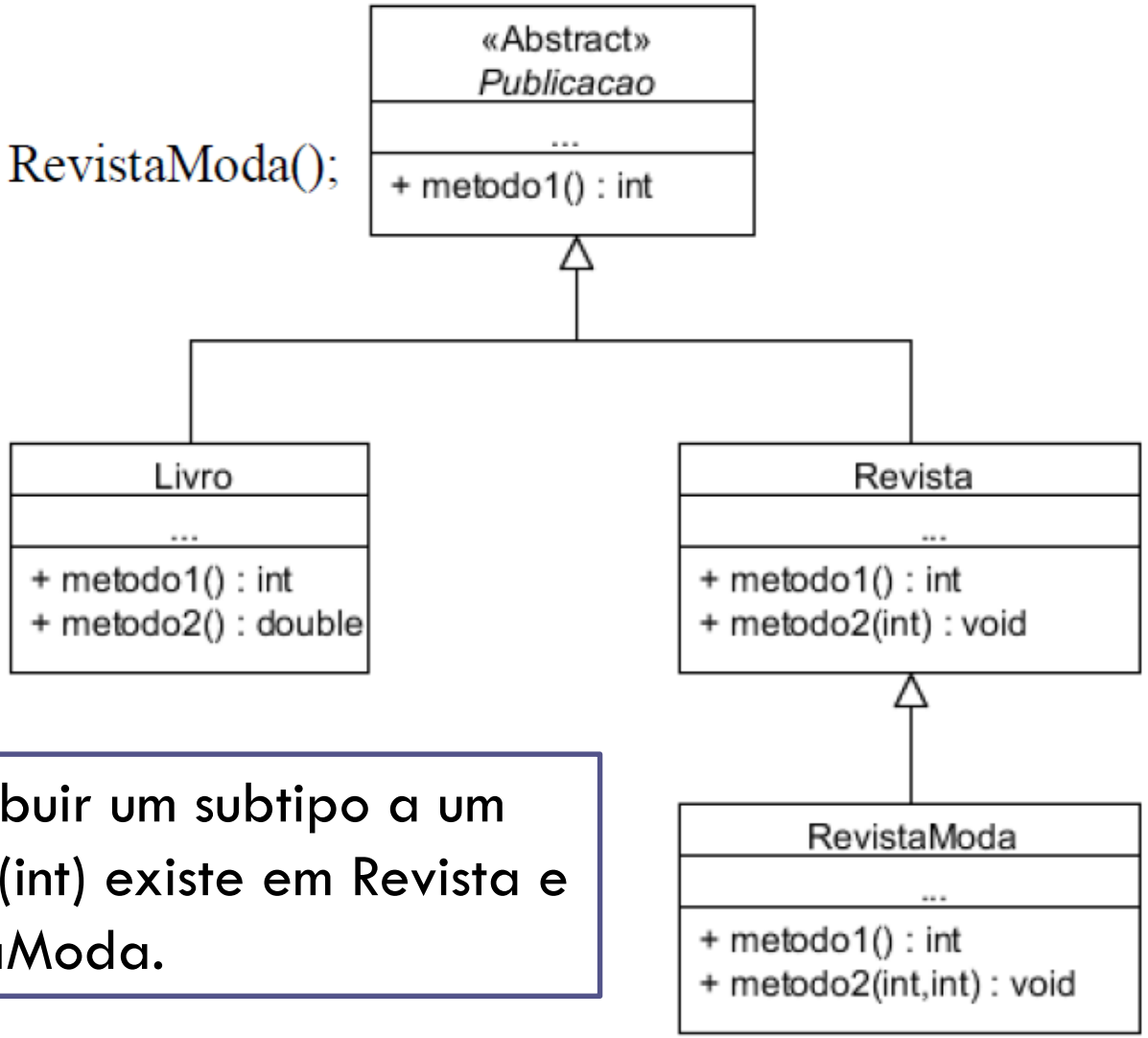


NÃO VÁLIDO. As primeiras 3 linhas estão corretas porque podemos atribuir um subtipo a um supertipo. No entanto, a 4ª linha dará erro porque metodo2() pertence apenas a Livro.

Grupo III

2. Para cada um dos blocos seguintes indique se o mesmo é ou não válido. Deverá justificar cada caso.

c.
```
Revista r;
RevistaModa rm = new RevistaModa();
r = rm;
r.metodo2(4);
```

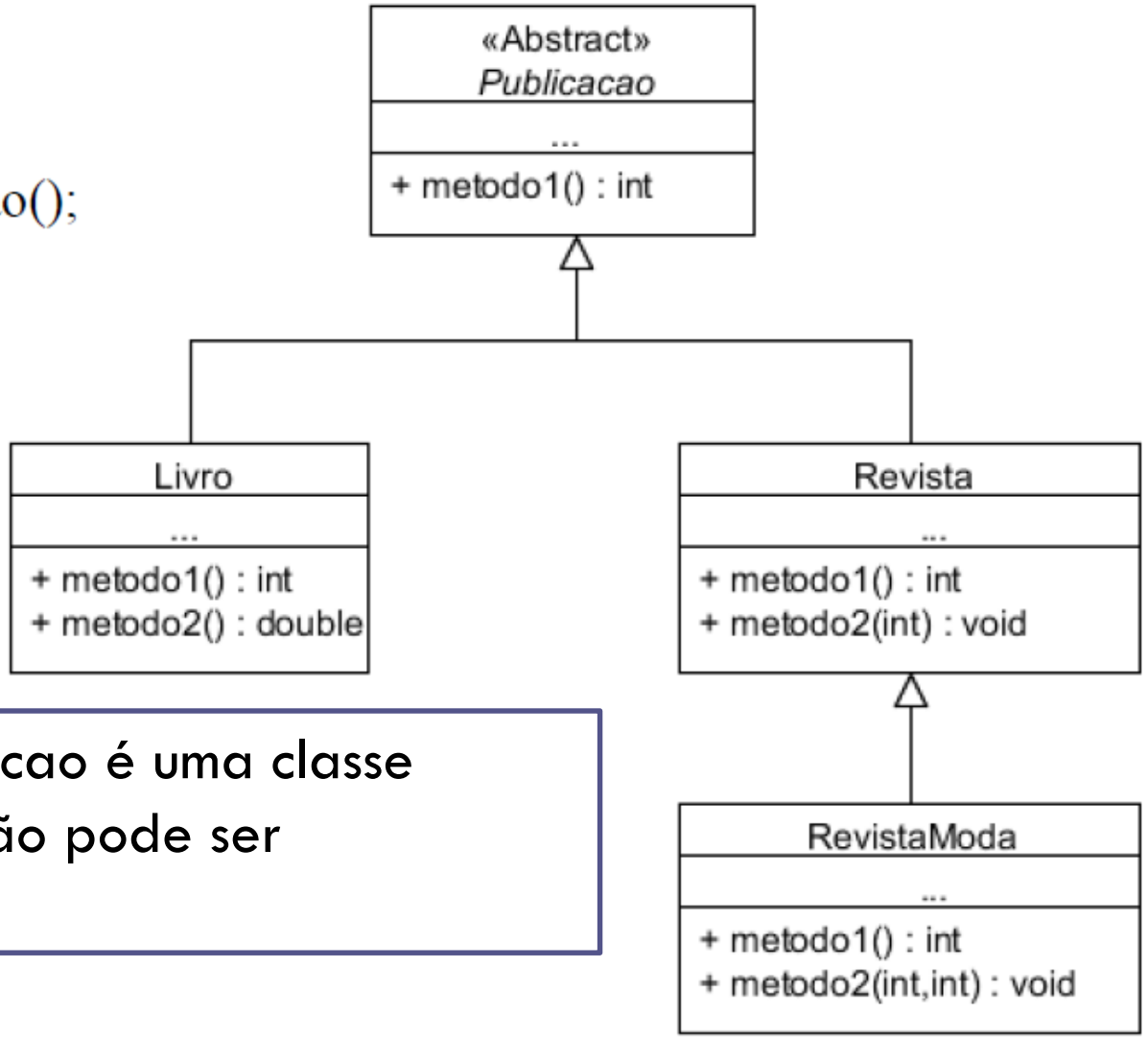


VÁLIDO. Podemos atribuir um subtipo a um supertipo. O metodo2(int) existe em Revista e é herdado por RevistaModa.

Grupo III

2. Para cada um dos blocos seguintes indique se o mesmo é ou não válido. Deverá justificar cada caso.

e.
```
Publicacao p1;
p2 = new Publicacao();
p1=p2;
p1.metodo1();
```



NÃO VÁLIDO. Publicacao é uma classe abstrata e como tal não pode ser instanciada.

Grupo IV

1. No código seguinte, que linhas darão erro de compilação?

```
1   public class X{
2     private int a;
3     public int b;
4     protected int c;
5
6     public X(){
7        a = 5;
8        b = 6;
9        c = 7;
10    }
11  }
12
13  public class Y extends X{
14    public Y(){
15      a = 3;
16      b = 4;
17      c = 5;
18    }
19  }
```

A - 7,15
B - 15
C - 15,17
D - 17
E - Nenhuma das anteriores

Grupo IV

2. Qual o resultado do código seguinte?

```
public class Teste{
   public static void main(String[] args){
      String[] frutos = {"Melao", "Uva",
              "Pera", "Melancia", "Kiwi"};
      System.out.print(frutos.length);
      System.out.print(frutos[2].length());
      System.out.print(frutos[3].charAt(1));
   }
}
```

A - 5 4 e
B - 5 3 M
C - 5 3 P
D - 4 4 e
E - Nenhuma das anteriores

Grupo IV

3. Qual o resultado do código seguinte?

```
String s1 = "Hello";
String s2 = "He"+"llo";
String s3 = new String("Hello");
System.out.print(s1==s2);
System.out.print(s1==s3);
System.out.print(s2.equals(s3));
```

A - true true true
B - true false true
C - true true false
D - false false true
E - Nenhuma das anteriores

Grupo IV

4. Qual o resultado do código seguinte se p.metodo() lançar uma Excecao2?

```
try {
      p.metodo();
}
catch (Excecao1 e) {
       System.out.print ("um");
}
catch (Excecao2 e) {
       System.out.print ("dois");
}
finally {
       System.out.print ("finalmente");
}
System.out.print ("acabou");
```

A - um dois finalmente acabou
B - dois finalmente acabou
C - finalmente acabou
D - dois acabou
E - um finalmente acabou

Grupo IV

5. No código seguinte, que linhas darão erro de compilação?

```
1   public class X{
2     public int metodo1(){ … }
3     public void metodo2(String a){ … }
4   }
5   public class Y extends X{
6     public int metodo1(){ … }
7     public void metodo3(int b){ … }
8   }
9   public class Teste{
10    public static void main(String[] args){
11      X v1 = new Y();
12      Y v2 = new Y();
13      v1.metodo3(5);
14      v2.metodo2("POO");
15    }
16  }
```

A - 13,14
B - 13
C - 14
D - 11
E - Nenhuma das anteriores