Dia 22 - Interfaces

CONTEUDO

O que é uma interface?

Como você já aprendeu, os objetos definem sua interação com o mundo exterior por meio dos métodos que expõem. Os métodos formam a interface do objeto com o mundo exterior; os botões na frente do seu aparelho de televisão, por exemplo, são a interface entre você e a fiação elétrica do outro lado de sua caixa de plástico. Você pressiona o botão "power" para ligar e desligar a televisão.

Em sua forma mais comum, uma interface é um grupo de métodos relacionados com corpos vazios. O comportamento de uma bicicleta, se especificado como uma interface, pode aparecer da seguinte forma:

```
interface Bicicleta {

    // rotações da roda por minuto
    void changeCadence (int newValue);

    void changeGear (int newValue);

    void speedUp (incremento interno);

    void applyBrakes (decremento interno);
}
```

Para implementar essa interface, o nome da sua classe mudaria (para uma determinada marca de bicicleta, por exemplo, como ACMEBicycle), e você usaria a palavra-chave implements na declaração da classe:

```
classe ACMEBicycle implementa Bicicleta {

    cadência int = 0;
    velocidade interna = 0;
    engrenagem interna = 1;
```

Dia 22 - Interfaces

```
    // O compilador agora exigirá que os métodos
    // changeCadence, changeGear, speedUp e applyBrakes
    // tudo seja implementado. A compilação irá falhar se aqueles
    // métodos estão faltando nesta classe.

     void changeCadence (int newValue) {
            cadência = novoValor;
     }

     void changeGear (int newValue) {
            gear = newValue;
     }

     void speedUp (int increment) {
            velocidade = velocidade + incremento;
     }

     void applyBrakes (int decrement) {
            velocidade = velocidade - decremento;
     }

     void printStates () {
            System.out.println ("cadência:" +
                  cadência + "velocidade:" +
                  velocidade + "engrenagem:" + engrenagem);
     }
}
```

Dia 22 - Interfaces

Implementar uma interface permite que uma classe se torne mais formal sobre o comportamento que promete fornecer. As interfaces formam um contrato entre a classe e o mundo externo, e esse contrato é aplicado no momento da construção pelo compilador. Se sua classe alega implementar uma interface, todos os métodos definidos por essa interface devem aparecer em seu código-fonte antes que a classe seja compilada com êxito

VÍDEO

https://www.youtube.com/watch?v=6uLLfRNgRA4&ab_channel=LoianeGroner

EXERCICIO

Com o seguinte codigo faça o exercicio 1, 2 e 3

```
package com.loiane.cursojava.aula44;

1.  public class Cachorro extends Mamifero implements AnimalDomesticado,
    AnimalEstimacao {

    a.    private int tamanho;
    b.    private String raça;

    c.    public int getTamanho() {
          i.      return tamanho;
    d.    }
    e.    public void setTamanho(int tamanho) {
          i.      this.tamanho = tamanho;
    f.    }
    g.    public String getRaça() {
          i.      return raça;
    h.    }
    i.    public void setRaça(String raça) {
          i.      this.raça = raça;
    j.    }
    k.    @Override
    l.    public void amamentar() {
          i.      // TODO Auto-generated method stub

    m.    }
    n.    @Override
    o.    public void emitirSom() {
          i.      // TODO Auto-generated method stub

    p.    }
    q.    @Override
    r.    public void levarVeterinario() {
          i.      // TODO Auto-generated method stub

    s.    }
    t.    @Override
    u.    public void alimentar() {
          i.      // TODO Auto-generated method stub
```

```
        v.     }
        w.     @Override
        x.     public void brincar() {
              i.       // TODO Auto-generated method stub

        y.     }
        z.     @Override
        aa.    public void levarPassear() {
              i.       // TODO Auto-generated method stub

        bb.    }


2.   }
```

==========

1. public abstract class Mamifero extends Animal {

    a. public abstract void amamentar();

2. }

==========

1. public abstract class Animal {

    a. private String nome;

    b. public abstract void emitirSom();

    c. public String getNome() {

        i. return nome;

    d. }

    e. public void setNome(String nome) {

        i. this.nome = nome;

    f. }

Dia 22 - Interfaces

2. }

==========

```
1.  public interface AnimalDomesticado {

      a.    public final int ANO = 2016;

      b.    void levarVeterinario();
      c.    void alimentar();
2.  }
```

==========

```
1.  public interface AnimalEstimacao {

      a.    void brincar();
      b.    void levarPassear();
2.  }
```

EXERCICIO

---

1)Qual é verdade

A. "X estende Y" está correto se e somente se X for uma classe e Y for uma interface

B. "X estende Y" está correto se e somente se X for uma interface e Y for uma classe

C. "X estende Y" está correto se X e Y forem ambas as classes ou ambas as interfaces

D. "X estende Y" está correto para todas as combinações de X e Y sendo classes e / ou interfaces

2)

Dado:

1. Dispositivo de implementos eletrônicos de classe pública

{public void doIt () {}}

2. A classe abstrata Phone1 estende Electronic {}

3. A classe abstrata Phone2 estende Electronic

4. {public void doIt (int x) {}}

5. A classe Phone3 estende o dispositivo de implementos eletrônicos

6. {public void doStuff () {}}

Dia 22 - Interfaces

7. {public void doIt () dispositivo de interface; }

Qual é o resultado? (Escolha todas as opções aplicáveis.)

A. Construção bem-sucedida

B. A compilação falha com um erro na linha 1

C. A compilação falha com um erro na linha 3

D. A compilação falha com um erro na linha 5

E. A compilação falha com um erro na linha 7

F. A compilação falha com um erro na linha 9

3) Dado:

interface pública abstrata Frobnicate {public void twiddle (String s); }

Qual é a classe correta? (Escolha todas as opções aplicáveis.)

A. public abstract class Frob implementa Frobnicate {

public abstract void twiddle (String s) {}

}

B. classe abstrata pública Frob implementa Frobnicate {}

C. public class Frob extends Frobnicate {

public void twiddle (Inteiro i) {}

}

D. public class Frob implementa Frobnicate {

public void twiddle (Inteiro i) {}

}

PROVA

---

1) Dado:

interface Hungry <E> {void munch (E x); }

interface Carnivore <E extends Animal> extends Hungry <E> {}

interface Herbívoro <E extends Plant> extends Hungry <E> {}

classe abstrata Plant {}

classe Grass extends Plant {}

Dia 22 - Interfaces

classe abstrata Animal {}

class Sheep extends Animal implementa Herbívoro <Sheep> {

público void munch (Carneiro x) {}

}

classe Wolf extends Animal implementa Carnivore <Sheep> {

público void munch (Carneiro x) {}

}

Qual das seguintes alterações (consideradas separadamente) permitiria a compilação desse código?

(Escolha todas as opções aplicáveis.)

A. Altere a interface do Carnivore para

interface Carnivore <E extends Plant> extends Hungry <E> {}

B. Altere a interface do herbívoro para

interface Herbivore <E extends Animal> extends Hungry <E> {}

C. Mude a classe Ovelha para

class Sheep extends Animal implementa Herbívoro <Planta> {

public void munch (Grass x) {}

}

D. Mude a classe Ovelha para

classe Carneiro extends Planta implementa Carnivore <Wolf> {

public void munch (Wolf x) {}

}

E. Altere a classe Wolf para

classe Wolf extends Animal implementa Herbívoro <Grass> {

public void munch (Grass x) {}

}

F. Nenhuma mudança é necessária

Dia 22 - Interfaces

2)

Dado:

1. Dispositivo de implementos eletrônicos de classe pública

{public void doIt () {}}

2. A classe abstrata Phone1 estende Electronic {}

3. A classe abstrata Phone2 estende Electronic

4. {public void doIt (int x) {}}

5. A classe Phone3 estende o dispositivo de implementos eletrônicos

6. {public void doStuff () {}}

7. {public void doIt () dispositivo de interface; }

Qual é o resultado? (Escolha todas as opções aplicáveis.)

A. Construção bem-sucedida

B. A compilação falha com um erro na linha 1

C. A compilação falha com um erro na linha 3

D. A compilação falha com um erro na linha 5

E. A compilação falha com um erro na linha 7

F. A compilação falha com um erro na linha 9

3) O seguinte bloco de código cria um Thread usando um destino Runnable:

Alvo executável = novo MyRunnable ();

Tópico meuThread = novo Tópico (destino);

Qual das seguintes classes pode ser usada para criar o destino, de modo que o código anterior compila corretamente?

A. public class MyRunnable estende Runnable {public void run () {}}

B. public class MyRunnable estende Object {public void run () {}}

C. public class MyRunnable implementa Runnable {public void run () {}}

D. public class MyRunnable implementa Runnable {void run () {}}

E. public class MyRunnable implementa Runnable {public void start () {}}