MC322 - Programação orientada a objetos
Aula 4.2
Enumerações

Prof. Marcos M. Raimundo
Instituto de Computação - UNICAMP

Enumerações

- Tipos enumerados
  - Um tipo enumerado é um tipo de dados especial que permite que uma variável assuma um conjunto de constantes pré-definidas
  - Como são constantes, os valores da enumeração deve estar escritos em maiúsculo
  - Exemplos:
    - indicações da bússola (NORTE, SUL, LESTE e OESTE)
    - dias da semana (SEGUNDA, TERÇA, QUARTA, QUINTA, SEXTA, SÁBADO E DOMINGO)
- Diferentemente de outras linguagens, as constantes de uma enumeração em Java não estão associadas a numeros.

Enumerações

- Devemos usar tipos enum quando queremos representar um conjunto fixo de constantes
- As constantes de uma enumeração são implicitamente finais e estáticas e podem ser usadas como rótulos de uma instrução switch
- Notação:

```
<modificador de acesso> enum MyEnum {
  CONSTANT1, ..., CONSTANTN
}
```

- Exemplo:

```
public enum Dia {
  SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO
}
```

Enumerações

```
public class EnumTest {
    Dia day;

    public EnumTest(Dia day){
        this.day = day;
    }

    public void tellItLikeItIs(){
        switch (day) {
            case SEGUNDA:
                System.out.println("Odeio segundas!")
                break;
            case SEXTA:
                System.out.println("Adoro a
                    sexta-feita.")
                break;
            case SABADO:
            case DOMINGO:
                System.out.println("A segunda já tá
                    batendo na porta. :-(")
                break;
            default:
                System.out.println("Esperando a sexta
                    ...")
                break;
        }
    }
```

```
    public static void main(String[] args){
        EnumTest firstDia = new EnumTest(Dia.SEGUNDA);
        firstDia.tellItLikeItIs();
        EnumTest firstDia = new EnumTest(Dia.QUARTA);
        firstDia.tellItLikeItIs();
        EnumTest firstDia = new EnumTest(Dia.SEXTA);
        firstDia.tellItLikeItIs();
        EnumTest firstDia = new EnumTest(Dia.SABADO);
        firstDia.tellItLikeItIs();
        EnumTest firstDia = new EnumTest(Dia.DOMINGO);
        firstDia.tellItLikeItIs();
    }

}
public enum Dia {
  SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO,
      DOMINGO
}
```

```
> Odeio segundas!
> Esperando a sexta ...
> Adoro sexta-feira.
> A segunda já tá batendo na porta. :-(
> A segunda já tá batendo na porta. :-(
```

Enumerações

- Toda enumeração possui um método estático values() que retorna um array com as constantes enumeradas na mesma ordem em que elas foram declaradas
- é comum combinar o método values() com o laço for aprimorado
- Quando uma constante enumerada é convertida para String, seu identificador é usado na representação da String

Enumerações: values()

```
public static void main(String[] args){
    System.out.println("**********************")
    System.out.println("Imprimindo todos os valores da
        enum Dia:")
    for (Dia d : Dia.values())
        System.out.println(d);
    System.out.println("**********************")
    EnumTest firstDia = new EnumTest(Dia.SEGUNDA);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.QUARTA);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.SEXTA);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.SABADO);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.DOMINGO);
    firstDia.tellItLikeItIs();
}
```

```
> **********************
> Imprimindo todos os valores da enum Dia:
> SEGUNDA
> TERCA
> QUARTA
> QUINTA
> SEXTA
> SABADO
> DOMINGO
> **********************
> Odeio segundas!
> Esperando a sexta ...
> Adoro sexta-feira.
> A segunda já tá batendo na porta. :-(
> A segunda já tá batendo na porta. :-(
```

Enumerações: values()

```
public static void main(String[] args){
    System.out.println("**********************")
    System.out.println("Imprimindo todos os valores da
        enum Dia:")
    for (int i=0; i<(Dia.values()).length; ++i)
        System.out.println((Dia.values())[i]);
    System.out.println("**********************")
    EnumTest firstDia = new EnumTest(Dia.SEGUNDA);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.QUARTA);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.SEXTA);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.SABADO);
    firstDia.tellItLikeItIs();
    EnumTest firstDia = new EnumTest(Dia.DOMINGO);
    firstDia.tellItLikeItIs();
}
```

```
> **********************
> Imprimindo todos os valores da enum Dia:
> SEGUNDA
> TERCA
> QUARTA
> QUINTA
> SEXTA
> SABADO
> DOMINGO
> **********************
> Odeio segundas!
> Esperando a sexta ...
> Adoro sexta-feira.
> A segunda já tá batendo na porta. :-(
> A segunda já tá batendo na porta. :-(
```

Enumerações: Classes

- Um tipo enumerado consiste em uma classe, portanto **pode** incluir construtores, atributos e métodos
- A declaração de uma enumeração consiste de duas partes:
  - declaração das constantes
  - declaração dos membros de classe (atributos, construtores e métodos)
- As constantes devem ser declaradas antes dos atributos e métodos
- Quando métodos e atributos forem declarados, a lista de constantes da enum deve terminar com um ponto-e-virgula (;)

Enumerações: Classes

- Uma constante de enumeração pode ser considerada como uma variável (estática e final) que referencia um objeto do tipo enumerado
  - A declaração de uma constante de enumeração corresponde à instanciação de um objeto, logo argumentos podem ser passados para o construtor
  - O objeto referenciado por uma constante mantém suas próprias cópias das variáveis de instância
  - O construtor chamado é escolhido de acordo com as mesmas regras de métodos sobrecarregados, ou seja, depende da lista de argumentos passados na chamada

Enumerações: Classes

```
public enum Planet {
    //Constantes. Cada constante corresponde a um
        objeto enumerado inicializado conforme o
        construtor
    MERCURIO (3.303e+23, 2.4397e6),
    VENUS (4.869e+24, 6.0518e6),
    TERRA (5.976e+24, 6.37814e6)
    MARTE (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27, 7.1492e7),
    SATURNO (5.688e26, 6.0268e7),
    URANO (8.686e+25, 2.5559e7),
    NETUNO (1.024e+26, 2.4746e7);

    //Membros da classe

    //sistema metrico
    private final double massa;
    private final double raio;

    public static final double G = 6.67300E-11;

    Planet(double massa, double raio){
        this.massa = massa;
        this.raio = raio;
    }
    private double massa(){
        return massa;
    }

    private double raio(){
        return massa;
    }

    double surfaceGravity(){
        return G * massa / (raio * raio);
    }

    double surfaceWeight(double outraMassa) {
        return outraMassa * surfaceGravity;
    }
}
```

Enumeração: Classes

```
public class EnumTestPlaneta {
  public static void main(String[] args){
    System.out.println("**********************")
    double pesoNaTerra = 10;
    double massa =
         pesoNaTerra/Planet.TERRA.surfaceGravity();
    for (Planet p : Planet.values())
      System.out.println("Seu peso em %s eh %fN \n",
           p, p.surfaceWeight(massa));
      System.out.println("**********************")
    }
}
```

```
**********************
Seu peso em MERCURIO eh 3.777576N
Seu peso em VENUS eh 9.049991N
Seu peso em TERRA eh 10.000000N
Seu peso em MARTE eh 3.787372N
Seu peso em JUPITER eh 25.305575N
Seu peso em SATURNO eh 10.660155N
Seu peso em URANO eh 9.051272N
Seu peso em NETUNO eh 11.383281N
**********************
```

Enumerações: Classes

- Os construtores de um tipo enumerado são implicitamente privados. é um erro de compilação declarar um construtor de uma enumeração com os modificadores de acesso public ou protected
- Um tipo enumerado não contém outras instâncias fora aquelas definidas por constantes
- Tentar instanciar explicitamente um objeto de tipo enumerado (usando a palavra-chave new) consiste em um erro de compilação

Enumeração: Classes

```
public class EnumTestPlaneta {
    public static void main(String[] args){
        Planet teste = new Planet(1,1);
        System.out.println("**********************")

        double pesoNaTerra = 10;
        double massa = pesoNaTerra/Planet.TERRA.surfaceGravity();
        for (Planet p : Planet.values())
            System.out.println("Seu peso em %s eh %fN \n", p, p.surfaceWeight(massa));
        System.out.println("**********************")
    }
}
```

```
EnumTestPlaneta.java.3: enum types may not be instantiated
        Planet teste = new Planet(1,1);
1 error
```

Enumeração: Classes

Atributos estáticos (com exceção de constantes primitivas) de um tipo enumerado não podem ser referenciados pelos construtores da enumeração

```
public enum Planet {
    MERCURIO (3.303e+23, 2.4397e6),
    VENUS (4.869e+24, 6.0518e6),
    TERRA (5.976e+24, 6.37814e6)
    MARTE (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27, 7.1492e7),
    SATURNO (5.688e26, 6.0268e7),
    URANO (8.686e+25, 2.5559e7),
    NETUNO (1.024e+26, 2.4746e7);

    //sistema metrico
    private final double massa;
    private final double raio;
    private final double conta;

    public static final double G = 6.67300E-11;

    Planet(double massa, double raio){
        this.massa = massa;
        this.raio = raio;
        conta = raio*TERRA.surfaceGravity();
    }
```

```
EnumTestPlaneta.java:18: ilegal reference to static
      field from initializer
         conta = raio*TERRA.surfaceGravity();
```

Enumeração: Classes

Atributos estáticos (com exceção de constantes primitivas) de um tipo enumerado não podem ser referenciados pelos construtores da enumeração

```
public enum Planet {
    MERCURIO (3.303e+23, 2.4397e6),
    VENUS (4.869e+24, 6.0518e6),
    TERRA (5.976e+24, 6.37814e6)
    MARTE (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27, 7.1492e7),
    SATURNO (5.688e26, 6.0268e7),
    URANO (8.686e+25, 2.5559e7),
    NETUNO (1.024e+26, 2.4746e7);

    //sistema metrico
    private final double massa;
    private final double raio;

    public static final double G = 6.67300E-11;

    Planet(double massa, double raio){
        this.massa = massa;
        this.raio = raio*G; //nao dá erro mas nao faz sentido
    }
```

Enumeração: Classes

Podemos chamar um método externo a Enum que altere o valor de um atributo final da Enum?

```
public enum Teste {
    CONSTANTE(1);

    private final int minhaPropriedade;

    Teste(int x) {
        minhaPropriedade=x;
    }

    public void setMinhaPropriedade(int x){
        minhaPropriedade=x;
    }

    public int getMinhaPropriedade(){
        return minhaPropriedade;
    }
```

Tentar atualizar uma variável final em um método que pode ser invocado n vezes causa um erro de compilação! Esta atribuição, no entanto, pode ser feita no construtor.

```
Teste.java:11: cannot assign a value to final variable
      minhaPropriedade
                  minhaPropriedade=x;
1 error
```

Enumeração: Classes

Dadas duas enum Teste e Teste2, com as mesmas constantes. Podemos comparar as duas?

```
public enum Teste {
    CONSTANTE(1);

    private final int minhaPropriedade;

    Teste(int x) {
        minhaPropriedade=x;
    }

    public int getMinhaPropriedade(){
        return minhaPropriedade;
    }

public enum Teste2 {
    CONSTANTE(2);

    private final int minhaPropriedade;

    Teste(int x) {
        minhaPropriedade=x;
    }

    public int getMinhaPropriedade(){
        return minhaPropriedade;
    }
```

```
class TesteEnum {
  public static void main(Strings args[]){
    if (Teste.CONSTANTE == Teste2.CONSTANTE){
        System.out.println("Ok!");
    }
  }
}
```

Não... seria o mesmo que tentar comparar dois objetos de classes distintas.

```
TesteEnum.java:3: incomparable type: Teste and Teste2
    if (Teste.CONSTANTE == Teste2.CONSTANTE){
1 error
```

Enumerações: Interfaces

- Todos os enums estendem implicitamente java.lang.Enum
- Como uma classe só pode estender uma classe, já que a linguagem Java não suporta herança múltipla de estado, um enum não pode estender mais nada
- Entretanto, ela pode implementar uma interface

Enumerações: Interfaces

```
public interface Volume{
    double calcVolume();
}
// Indicacao que iremos implementar Volume
public enum Planet implements Volume {
    MERCURIO (3.303e+23, 2.4397e6),
    VENUS (4.869e+24, 6.0518e6),
    TERRA (5.976e+24, 6.37814e6)
    MARTE (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27, 7.1492e7),
    SATURNO (5.688e26, 6.0268e7),
    URANO (8.686e+25, 2.5559e7),
    NETUNO (1.024e+26, 2.4746e7);

    //Membros da classe

    //sistema metrico
    private final double massa;
    private final double raio;

    public static final double G = 6.67300E-11;

    Planet(double massa, double raio){
        this.massa = massa;
        this.raio = raio;
    }
```

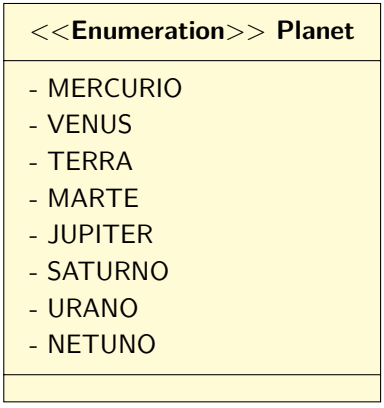
```
    private double massa(){
        return massa;
    }

    private double raio(){
        return massa;
    }

    double surfaceGrativy(){
        return G * massa / (raio * raio);
    }

    double surfaceWeight(double outraMassa) {
        return outraMassa * surfaceGravity;
    }

    // Implementacao do metodo Volume
    public double calcVolume(){
        return (Math.pow(raio, 3) * 4 * Math.PI/3);
    }
}
```

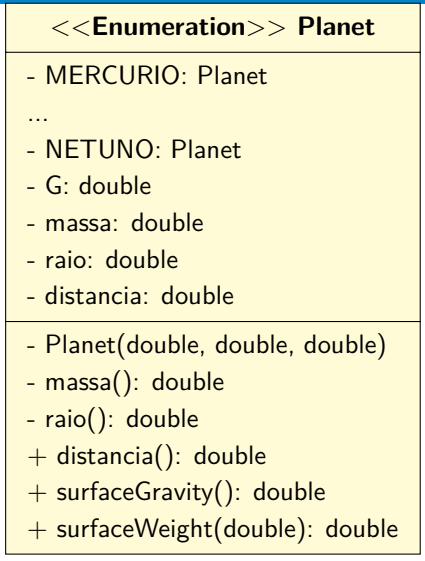
Enumerações: Interfaces

```
public class Esfera implements Volume{
    private double raio;

    public Esfera(double raio){
        this.raio = raio;
    }

    public double calcArea(){
        return (Math.pow(raio, 2) * 4 * Math.PI);
    }

    public double calcVolume(){
        return (Math.pow(raio, 3) * 4 * Math.PI)/3;
    }
}
```

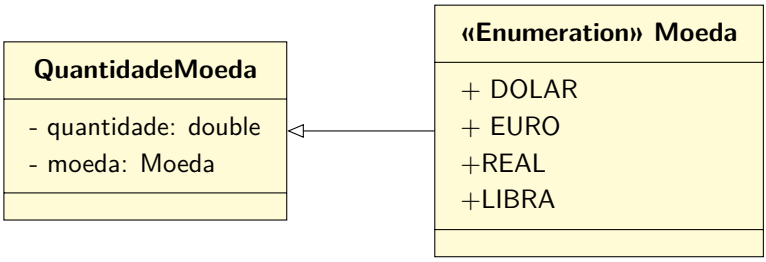
```
public class EnumEsfPlaneta{
  public static void main(String[] args){
  System.out.println("**********************");
  for (Volume v : Planet.values())
    System.out.printf("O volume do planet %s eh %f
        m3\n", v, v.calcVolume());
  System.out.println("**********************");
  Volume v = new Esfera(10);
  System.out.printf("O volume da esfera eh %g m3\n",
      v.calcVolume());
  System.out.printf("A area da esfera eh %f m2\n",
      ((Esfera) (v)).calcArea());
  }
}
```

Enumerações: UML





Enumerações: UML



Atividades Extra-classe

- Proponha a implementação das cartas de sorte/reves do banco imobiliário através de enum.
- Como seria a implementação dos elementos do tabuleiro do banco imobiliário se usássesmos enums?

Referências

- Java: Como Programar, Paul Deitel Heivey Deitel; Pearson; 7a. Ed.
- Object-Oriented Programming with Java: An Introduction, David J. Barnes; Prentice Hall (2000).
- Usberti, F. Notas de Aula de MC322.
- http://docs.oracle.com/javase/tutorial/

MC322 - Programação orientada a objetos
Aula 4.2
Enumerações

Prof. Marcos M. Raimundo
Instituto de Computação - UNICAMP