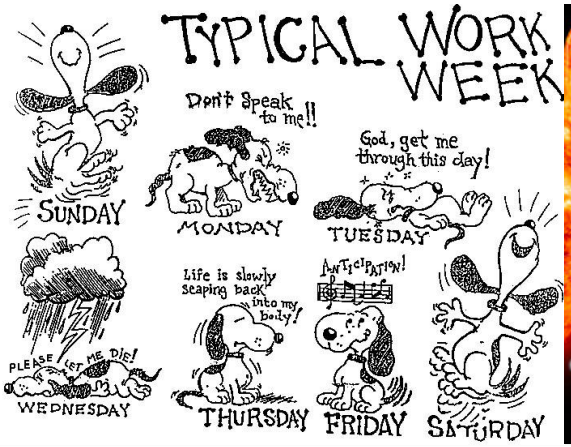
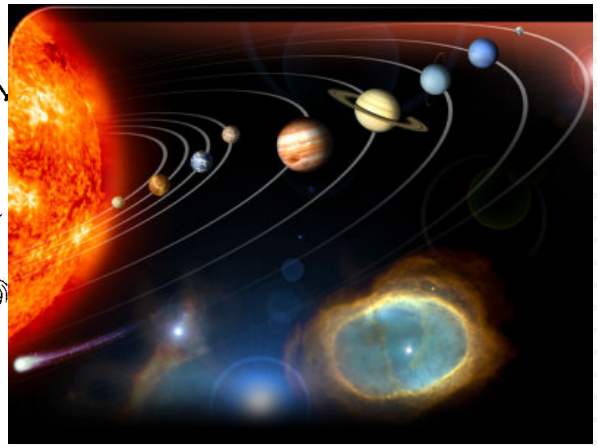
PROGRAMAÇÃO ORIENTADA PELOS OBJECTOS

Enumerações

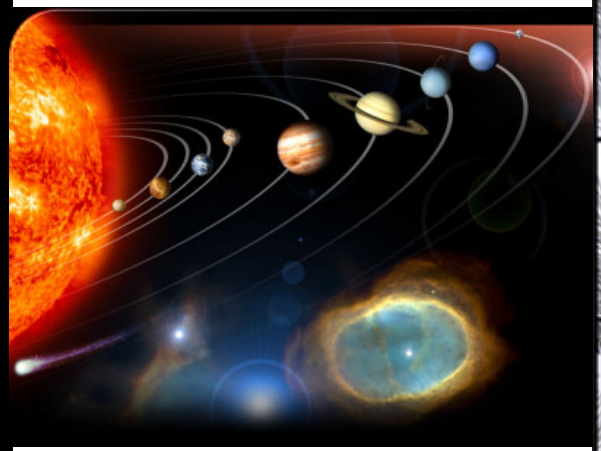
Tipos de dados Enumerados





Para que serve um tipo de dados enumerado?

- Construa uma aplicação que indica o peso de um corpo nos vários planetas do sistema solar



Fed up with how her diet is going, Charlene takes a more serious aim at her target weight.

Tipos enumerados

- Um tipo enumerado é um tipo cujos membros de dados são um conjunto fixo de constantes
- Exemplos típicos
  - Pontos cardeais
    - NORTH, SOUTH, EAST, WEST
  - Dias da semana
    - SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
  - Naipes de um baralho de cartas
    - CUBS, HEARTS, DIAMONDS, SPADES
- Nota: representamos os campos do tipo enumerado com maiúsculas, dado que se tratam de **constantes** – mantemos assim a convenção habitual para as constantes

Definição de um tipo enumerado

- Define-se um tipo enumerado com a palavra reservada enum. Exemplos:
  - Pontos cardeais:

```
public enum CompassDirections {
   NORTH, SOUTH, EAST, WEST
}
```

  - Dias da semana:

```
public enum WeekDay {
   SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}
```

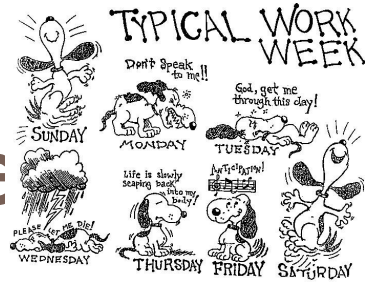
  - Naipes:

```
public enum CardSuit {
   CUBS, HEARTS, DIAMONDS, SPADES
}
```

Quando usar tipos enumerados?

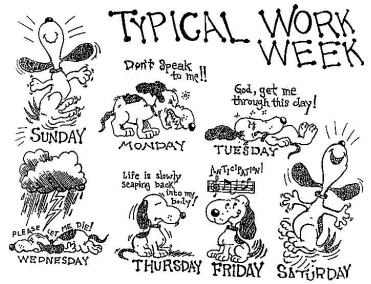
- Sempre que necessitamos de representar um conjunto fixo de constantes
  - Tipos enumerados naturais
    - tais como os pontos cardeais, dias da semana, meses do ano, ou planetas do sistema solar
  - Conjuntos de dados para os quais TODOS os valores possíveis são conhecidos em tempo de compilação
    - Escolhas de um menu, flags na linha de comando, etc.

Programando a classe de teste




```
public class Main {
  public static void tellItLikeItIs(WeekDay day) {
    System.out.print(day + " ");
    switch (day) {
      case TUESDAY:
      case FRIDAY:
        System.out.println("Maravilha, aula teórica de POO. ;-)");
        break;
      case MONDAY:
      case WEDNESDAY:
      case THURSDAY:
        System.out.println("Hoje é dia de Eclipse. Vampiros?!? :-[");
        break;
      case SATURDAY:
      case SUNDAY: System.out.println("Mais tempo para programar! :-)");
        break;
      default: System.out.println("Mas… há outros?");
        break;
    } // switch
  } // tellItLikeItIs
```

Programa principal

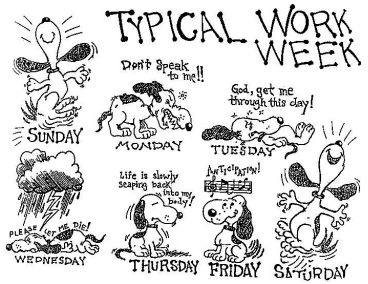



```
  public static void main(String[] args) {
    tellItLikeItIs(WeekDay.MONDAY);
    tellItLikeItIs(WeekDay.TUESDAY);
    tellItLikeItIs(WeekDay.WEDNESDAY);
    tellItLikeItIs(WeekDay.THURSDAY);
    tellItLikeItIs(WeekDay.FRIDAY);
    tellItLikeItIs(WeekDay.SATURDAY);
    tellItLikeItIs(WeekDay.SUNDAY);
  }
} // Fim da classe EnumTest
```

Isto assim é muito repetitivo.
Podemos fazer melhor!

```
MONDAY Hoje é dia de Eclipse. Vampiros?!? :-[
TUESDAY Maravilha, aula teórica de POO ;-)
WEDNESDAY Hoje é dia de Eclipse. Vampiros?!? :-[
THURSDAY Hoje é dia de Eclipse. Vampiros?!? :-[
FRIDAY Maravilha, aula teórica de POO ;-)
SATURDAY Fim de semana! Mais tempo para programar! :-)
SUNDAY Fim de semana! Mais tempo para programar! :-)
```

Programa principal




```
  public static void main(String[] args) {
    for(WeekDay d: WeekDay.values())
      tellItLikeItIs(d);
  }
} // Fim da classe EnumTest
```

A operação values devolve um vector de WeekDay. Esta operação é herdada da classe

java.lang.enum

O ciclo for each visita todos os elementos do tipo enumerado.

```
MONDAY Hoje é dia de Eclipse. Vampiros?!? :-[
TUESDAY Maravilha, aula teórica de POO ;-)
WEDNESDAY Hoje é dia de Eclipse. Vampiros?!? :-[
THURSDAY Hoje é dia de Eclipse. Vampiros?!? :-[
FRIDAY Maravilha, aula teórica de POO ;-)
SATURDAY Fim de semana! Mais tempo para programar! :-)
SUNDAY Fim de semana! Mais tempo para programar! :-)
```

O que é, afinal, um tipo enumerado em Java?

- Um tipo enumerado define uma classe que herda de java.lang.enum
- A declaração do enumerado pode incluir métodos e outros campos
- Por ser sub-classe de java.lang.enum, tem automaticamente algumas operações
  - values() devolve um vector com todos os valores do tipo enumerado, pela ordem com a qual foram declarados
  - Este método é normalmente usado em conjugação com o for-each

Exemplo: Planetas


```
public enum Planet {

  private final double mass; // in kilograms
  private final double radius; // in meters

  private Planet(double mass, double radius) {
    this.mass = mass;
    this.radius = radius;
  }
```

Exemplo: Planetas


```
public enum Planet {
  // Planet (double mass, double radius)
  MERCURY (3.303e+23, 2.4397e6),
  VENUS   (4.869e+24, 6.0518e6),
  EARTH   (5.976e+24, 6.37814e6),
  MARS    (6.421e+23, 3.3972e6),
  JUPITER (1.9e+27,   7.1492e7),
  SATURN  (5.688e+26, 6.0268e7),
  URANUS  (8.686e+25, 2.5559e7),
  NEPTUNE (1.024e+26, 2.4746e7);

  private final double mass; // in kilograms
  private final double radius; // in meters

  private Planet(double mass, double ra
    this.mass = mass;
    this.radius = radius;
  }
```

Cada constante do enumerado Planet é declarada com valores para os parâmetros da massa e raio dos planetas.
Estes valores são passados ao **private** construtor quando a constante é criada.
As constantes têm de ser declaradas antes de quaisquer outros membros de dados, ou operações.

O construtor tem de ser privado, ou de visibilidade package. Não pode ser chamado de fora. Cria automaticamente as constantes declaradas no início do corpo da declaração do tipo enumerado.

Exemplo: Planetas


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

  private double radius() {
    return radius;
  }

  // universal gravitational constant  (m3 kg-1 s-2)
  public static final double G = 6.67300E-11;

  public double surfaceGravity() {
    return G * mass() / (radius() * radius());
  }

  public double surfaceWeight(double otherMass) {
    return otherMass * surfaceGravity();
  }
}
```

O tipo enumerado contém outras operações e mesmo uma constante G. Note que o valor de G, apesar de constante, **não é** um dos elementos do tipo enumerado.

Exemplo: Planetas


```
public class Main {

/**
 * Este programa permite calcular o peso de uma pessoa em cada um
 * dos planetas do sistema solar.
 * @param args
 */
    public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      double earthWeight = in.nextDouble();
      double mass = earthWeight/Planet.EARTH.surfaceGravity();
      for (Planet p : Planet.values())
        System.out.printf("Your weight on %s is %f%n",
                          p, p.surfaceWeight(mass));
    }
}
```

Exemplo de interacção

- Exemplo de cálculo do peso de uma pessoa com 70kg (na terra), nos vários planetas do sistema solar

```
70
Your weight on MERCURY is 26.443033
Your weight on VENUS is 63.349937
Your weight on EARTH is 70.000000
Your weight on MARS is 26.511603
Your weight on JUPITER is 177.139027
Your weight on SATURN is 74.621088
Your weight on URANUS is 63.358904
Your weight on NEPTUNE is 79.682965
```

16

Comandos como Enumerados

Enumerado Command


```
public class Main {
   // Comandos do utilizador
   private enum Command {
        QUIT, REGISTA, CONSULTAPESSOA, AMIGOS, CONSULTAMIGOS,
        CONSULTAESTADO, NOVOESTADO, PESSOAS, UNKNOWN
   };
   ...

   private static Command getCommand(Scanner input) {
     try {
       String comm = input.nextLine().toUpperCase();
       return Command.valueOf(comm);
     } catch (IllegalArgumentException e) {
     // se o comando não for reconhecido
     return UNKNOWN;
   }
}
```

Enumerado Command


```
private static void commands() {
   SocialNetwork sn = new SocialNetworkClass();
   Scanner in = new Scanner(System.in);
   Command c = getCommand(in);
   while (!c.equals(QUIT)) {
        switch (c) {
          case REGISTA: registerPerson(in, sn); break;
          case CONSULTAPESSOA : hasPerson(in, sn); break;
          case AMIGOS: setFriendship(in, sn); break;
          case CONSULTAMIGOS : numberOfFriends(in, sn); break;
          case CONSULTAESTADO : getStatus(in, sn); break;
          case NOVOESTADO : setStatus(in, sn); break;
          case PESSOAS : list(sn); break;
          default: // Não faz nada
          break;
        }
        c = getCommand(in);
   } System.out.println(BYE);
}
```