Enumerated Types

CSE 114: Introduction to Object-Oriented Programming

Paul Fodor

Stony Brook University

http://www.cs.stonybrook.edu/~cse114

Enumerated Types

- An enumerated type defines a list of enumerated values
  - Each value is an identifier

```
enum MyFavoriteColor {RED, BLUE, GREEN, YELLOW};
```

    - A value of an enumerated type is like a constant and so, by convention, is spelled with all uppercase letters
    - Also, by convention, an enumerated type is named like a class with first letter of each word capitalized
  - Once a type is defined, you can declare a variable of that type:

```
MyFavoriteColor color;
```

    - The variable **`color`** can hold one of the values defined in the enumerated type **`MyFavoriteColor`** or **`null`**, but nothing else
- Using enumerated values (e.g., **`Color.BLUE`**, **`Day.MONDAY`**) rather than literal integer values (e.g., **`0`**, **`1`**, and so on) can make program easier to read and maintain

Enumerated Types

- The enumerated values can be accessed using the syntax

`EnumeratedTypeName.valueName`

- For example, the following statement assigns enumerated value **`BLUE`** to variable **`color`**:

```
color = MyFavoriteColor.BLUE;
```

- An enumerated type is treated as a special class, so an enumerated type variable is therefore a reference variable
  - An enumerated type is a subtype of the **`Object`** class (inherits all the methods in the **`Object`** class) and the **`Comparable`** interface (has the **`compareTo`** method in the **`Comparable`** interface)

Enumerated Types

- The following methods are defined for any enumerated object:

```
public String name();
```

  - Returns a name of the value for the object

```
public int ordinal();
```

  - Returns the ordinal value associated with the enumerated value
  - The first value in an enumerated type has an ordinal value of 0, the second has an ordinal value of 1, the third one 3, and so on

```
public class EnumeratedTypeDemo {
  static enum Day {SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
    FRIDAY, SATURDAY};

  public static void main(String[] args) {
    Day day1 = Day.FRIDAY;
    Day day2 = Day.THURSDAY;

    System.out.println("day1's name is " + day1.name());
    System.out.println("day2's name is " + day2.name());

    System.out.println("day1's ordinal is " + day1.ordinal());
    System.out.println("day2's ordinal is " + day2.ordinal());

    System.out.println("day1.equals(day2) returns " +
      day1.equals(day2));
    System.out.println("day1.toString() returns " +
      day1.toString());
    System.out.println("day1.compareTo(day2) returns " +
      day1.compareTo(day2));
  }
}
```

```
enum Day {SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
    FRIDAY, SATURDAY};

public class EnumeratedTypeDemo {
  public static void main(String[] args) {
    Day day1 = Day.FRIDAY;
    Day day2 = Day.THURSDAY;

    System.out.println("day1's name is " + day1.name());
    System.out.println("day2's name is " + day2.name());

    System.out.println("day1's ordinal is " + day1.ordinal());
    System.out.println("day2's ordinal is " + day2.ordinal());

    System.out.println("day1.equals(day2) returns " +
      day1.equals(day2));
    System.out.println("day1.toString() returns " +
      day1.toString());
    System.out.println("day1.compareTo(day2) returns " +
      day1.compareTo(day2));
  }
}
```

```
day1's name is FRIDAY
day2's name is THURSDAY
day1's ordinal is 5
day2's ordinal is 4
day1.equals(day2) returns false
day1.toString() returns FRIDAY
day1.compareTo(day2) returns 1
```

Enumerated Types

- An enumerated type can be defined inside a class or standalone
  - After the first program is compiled, a class named **`EnumeratedTypeDemo$Day.class`** is created
    - When an enumerated type is declared inside a class, the type must be declared as a **`static`** member of the class and cannot be declared inside a method
    - **`static`** may be omitted
  - In the latter case, the type is treated as a standalone class, so after the program is compiled, a class named **`Day.class`** is created

Using if or switch Statements with an Enumerated Variable

- Often your program needs to perform a specific action depending on the value
  - For example, if the value is **`Day.MONDAY`**, play soccer; if the value is **`Day.TUESDAY`**, take piano lesson, and so on

```
if (day.equals(Day.MONDAY)) {
    // process Monday
} else if (day.equals(Day.TUESDAY)) {
    // process Tuesday
} else
    ...
```

Using if or switch Statements with an Enumerated Variable

```
switch (day) {
    case MONDAY:
         // process Monday
         break;
    case TUESDAY:
         // process Tuesday
         break;
    ...
}
```

- In the switch statement, the case label is an unqualified enumerated value (e.g., **MONDAY**, but not **Day.MONDAY**).

Processing Enumerated Values Using a Foreach Loop

- Each enumerated type has a static method **`values()`** that returns all enumerated values for the type in an array:

```
Day[] days = Day.values();
for (int i = 0; i < days.length; i++)
 System.out.println(days[i]);
// is equivalent with:
for (Day day: days)
 System.out.println(day);
```

Enumerated Types with Data Fields, Constructors, and Methods

```
public enum TrafficLight {
  RED ("Please stop"), GREEN ("Please go"),
    YELLOW ("Please caution");

  private String description;
  private TrafficLight(String description) {
    this.description = description;
  }
  public String getDescription() {
    return description;
  }
};
```

- The constructor is invoked whenever an enumerated value is accessed
  - The enumerated value's argument is passed to the constructor, which is then assigned to **`description`**

Enumerated Types with Data Fields, Constructors, and Methods

```
public class TestTrafficLight {
  public static void main(String[] args) {
    TrafficLight light = TrafficLight.RED;
    System.out.println(light.getDescription());
  }
}
```

- An enumerated value **`TrafficLight.RED`** is assigned to variable **`light`**
- Accessing **`TrafficLight.RED`** causes the JVM to invoke the constructor with argument "**`please stop`**"