Working with Enums

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim   jwhh.com

Overview

The role of enum types

Declaring enum types

Conditional logic with enums

Relative comparisons of enums

Common enum methods

Using class-based features of enums

```java
public enum FlightCrewJob {
   FLIGHT_ATTENDANT,
   COPILOT,
   PILOT
}
```

Enumeration Types

Useful for defining a type with a finite list of valid values

- Declare using the enum keyword
- Provide comma-separated value list
- By convention value names are all upper case

```java
FlightCrewJob job1 = FlightCrewJob.PILOT;

FlightCrewJob job2 = FlightCrewJob.FLIGHT_ATTENDANT;

if(job1 == FlightCrewJob.PILOT)
    System.out.println("job1 is PILOT");

if(job1 != job2)
    System.out.println("job1 and job2 are different");
```

Conditional Logic

Enums support equality tests

- Can use == and != operators

Enums support switch statements

Main.java

```java
void displayJobResponsibilities(FlightCrewJob job) {
  switch(job) {
    case FLIGHT_ATTENDANT:
      System.out.println("Assures passenger safety");
      break;
    case COPILOT:
      System.out.println("Assists in flying the plane");
      break;
    case PILOT:
      System.out.println("Flies the plane");
      break;
  }
}
```

Relative Comparisons

Values are ordered

- First value is lowest
- Last value is highest

Use compareTo for relative comparison

- Returns negative, zero, or positive value
- Indicates current instance’s ordering relative to another value

Relative Comparisons

FlightCrewJob.java

```java
public enum FlightCrewJob {
    FLIGHT_ATTENDANT,
    COPILOT,
    PILOT
}
```

CrewMember.java

```java
class CrewMember {
  FlightCrewJob job;
  String name;
  CrewMember(FlightCrewJob job,
             String name) {
    this.job = job;
    this.name = name;
  }
} // other members elided
```

Main.java

```java
CrewMember geetha = new CrewMember(FlightCrewJob.PILOT, "Geetha");
CrewMember bob = new CrewMember(FlightCrewJob.FLIGHT_ATTENDANT, "Bob");
whoIsInCharge(geetha, bob);

void whoIsInCharge(CrewMember member1, CrewMember member2)
  CrewMember theBoss = member1.getJob().compareTo(member2.getJob()) > 0
          member1 : member2;
  System.out.println(theBoss.getName() + " is boss"); // Geetha is boss
}
```

Common Enum Methods

Method	Description
values	Returns an array contain all values
valueOf	Returns the value that corresponds to a string (case sensitive)

Enum Types Are Classes

Implicitly inherit from Java’s Enum class

Similar to other class in some ways

Have some special characteristics

Enum Types Can Have Members

Fields

Methods

Constructors

Enum Values

Each value is an instance of the enum type

Declaring the value creates the instance

Can leverage the enum type’s constructor

FlightCrewJob.java

```
public enum FlightCrewJob {
  FLIGHT_ATTENDANT,
  COPILOT,
  PILOT;

  private String title;
  public String getTitle() { return title; }
  private FlightCrewJob(String title) {
    this.title = title;
  }
}
```

FlightCrewJob.java

```java
public enum FlightCrewJob {
  FLIGHT_ATTENDANT,
  COPILOT,
  PILOT("Captain");

  private String title;
  public String getTitle() { return title; }
  private FlightCrewJob(String title) {
    this.title = title;
  }
}
```

FlightCrewJob.java

```java
public enum FlightCrewJob {
  FLIGHT_ATTENDANT,
  COPILOT("First Officer"),
  PILOT("Captain");

  private String title;
  public String getTitle() { return title; }
  private FlightCrewJob(String title) {
    this.title = title;
  }
}
```

FlightCrewJob.java

```java
public enum FlightCrewJob {
  FLIGHT_ATTENDANT("Flight Attendant"),
  COPILOT("First Officer"),
  PILOT("Captain");

  private String title;
  public String getTitle() { return title; }
  private FlightCrewJob(String title) {
    this.title = title;
  }
}
```

Main.java

```java
CrewMember geetha = new CrewMember(FlightCrewJob.PILOT, "Geetha");
CrewMember bob = new CrewMember(FlightCrewJob.FLIGHT_ATTENDANT, "Bob");
whoIsInCharge(geetha, bob);


void whoIsInCharge(CrewMember member1, CrewMember member2)
  CrewMember theBoss = member1.getJob().compareTo(member2.getJob()) > 0
          member1 : member2;
  System.out.println(theBoss.getJob().getTitle() + " " +
        theBoss.getName() + " is boss"); // Geetha is boss
}
```

Main.java

```
CrewMember geetha = new CrewMember(FlightCrewJob.PILOT, "Geetha");
CrewMember bob = new CrewMember(FlightCrewJob.FLIGHT_ATTENDANT, "Bob");
whoIsInCharge(geetha, bob);


void whoIsInCharge(CrewMember member1, CrewMember member2)
  CrewMember theBoss = member1.getJob().compareTo(member2.getJob()) > 0
          member1 : member2;
  System.out.println(theBoss.getJob().getTitle() + " " +
        theBoss.getName() + " is boss"); // Captain Geetha is boss
}
```

Summary

Enumeration types

- Define a finite list of valid values

Support conditional logic

- Can perform equality tests
- Work well with switch statements

Enum values are ordered

- First value is lowest
- Last value is highest
- Can perform order-based comparisons with compareTo

Summary

Enum types are classes

- Inherit from Java's Enum class
- Can define members

Enum values

- Are instances of the enum type
- Declaring a value creates the instance
- Can leverage constructors