Enum Types

Exercises and solutions

1. What are enum types in Java?

**Answer:**

An enum or enumeration or enumerated data type lets user create an ordered list of constants as a type. It defines a new reference type in Java.

1. What is the superclass of all enums in Java?

**Answer:**

Every enum type is implicitly inherited from the java.lang.Enum class

1. Can an enum in Java extend another enum?

**Answer:**

No

1. Can an enum in Java implement one or more interfaces?

**Answer:**

Yes

1. Is the following enum declaration valid? If yes, how many enum constants does it declare?  
     
   public enum Gender {  
    MALE, FEMALE,   
   }

**Answer:**

Yes. The Gender enum declares two constants, MALE and FEMALE.

1. Consider the following declaration for an enum named Day.  
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }  
     
   Given a string "FRIDAY", how will you look up the Day.FRIDAY enum constant?

**Answer:**

Day friday = Day.valueOf("FRIDAY");

1. Consider the following declaration for an enum named Day.  
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }  
     
   How will you look up the ordinal for Day.SUNDAY?

**Answer:**

int sundayOrdinal = Day.SUNDAY.ordinal();

1. Consider the following declaration for an enum named Day.  
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }

Complete the following snippet of code that will print the ordinal of TUESDAY from the Day enum. It should print 1.  
  
String dayName = "TUESDAY";  
int ordinal = /\* Complete this statement. \*/;  
System.out.println(ordinal);

**Solution:**

String dayName = "TUESDAY";

int ordinal = Day.valueOf(dayName).ordinal();

System.out.println(ordinal);

1. Consider the following declaration for an enum named Day.  
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }  
     
   Use a for-each loop to print name of each day with its ordinal like MONDAY(0), TUESDAY(1), etc.

**Solution:**

for(Day day : Day.values()) {

String name = day.name();

int ordinal = day.ordinal();

System.out.println(name + "(" + ordinal + ")");

}

1. Write the output for the following snippet of code.  
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }  
     
   EnumSet<Day> es = EnumSet.range(Day.TUESDAY, Day.FRIDAY);  
   for(Day d : es) {  
    System.out.printf("%s(%d)%n", d.name(), d.ordinal());  
   }

**Answer:**

TUESDAY(1)

WEDNESDAY(2)

THURSDAY(3)

FRIDAY(4)

1. Write the output for the following snippet of code.  
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }  
     
   EnumSet<Day> es = EnumSet.complementOf(EnumSet.range(Day.TUESDAY, Day.FRIDAY));  
   for(Day d : es) {  
    System.out.printf("%s(%d)%n", d.name(), d.ordinal());  
   }

**Answer:**

MONDAY(0)

SATURDAY(5)

SUNDAY(6)

1. Consider the following declaration for an enum named Country.   
     
   public enum Country {  
    BHUTAN("Bhutan", "BT"),   
    BRAZIL("Brazil", "BR"),   
    FIJI("Fiji", "FJ"),  
    INDIA("India", "IN"),  
    SPAIN("Spain", "ES");  
     
    private final String fullName;  
    private final String isoName;  
     
    private Country(String fullName, String isoName) {  
    this.fullName = fullName;  
    this.isoName = isoName;  
    }  
     
    public String fullName() {  
    return this.fullName;  
    }  
     
    public String isoName() {  
    return this.isoName;  
    }  
     
    @Override  
    public String toString() {  
    return this.fullName;  
    }   
   }  
     
   Write the output when the following snippet of code is executed:  
     
   for(Country c : Country.values()) {  
    System.out.printf("%s[%d, %s, %s]%n",   
    c.name(), c.ordinal(), c, c.isoName());  
   }

**Answer:**:

BHUTAN[0, Bhutan, BT]

BRAZIL[1, Brazil, BR]

FIJI[2, Fiji, FJ]

INDIA[3, India, IN]

SPAIN[4, Spain, ES]

1. Consider the following declaration for a Gender enum:  
     
   public enum Gender {  
    MALE, FEMALE   
   }  
     
   Modify the code for the Gender enum, so that the output of the following snippet of code is as shown in the expected output section that follows the code. You are supposed to change the code for the Gender enum, not the following snippet of code.  
     
   for(Gender c : Gender.values()) {  
    System.out.printf("%s%n", c);  
   }  
     
   Expected output:  
   male  
   female

**Solution:**

public enum Gender {

MALE, FEMALE;

@Override

public String toString() {

return this.name().toLowerCase();

}

}

1. Suppose Color is an enum. Is the following declaration of a MyFavColor enum valid? If not, explain your answer.  
     
   public enum MyFavColor extends Color {  
    WHITE, BLACK  
   }

**Answer:**

The declaration for the MyFavColor enum is invalid. An enum cannot extend another enum or class.

1. What will be the output when the following snippet of code is run?   
     
   public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
   }  
     
   Day[] days = {Day.FRIDAY, Day.MONDAY, Day.WEDNESDAY};  
   System.out.println(Arrays.toString(days));  
   Arrays.sort(days);  
   System.out.println(Arrays.toString(days));

**Answer:**

[FRIDAY, MONDAY, WEDNESDAY]

[MONDAY, WEDNESDAY, FRIDAY]

1. What will be the output of the following snippet of code?  
     
   public enum Gender {  
    MALE, FEMALE  
   }  
     
   System.out.println(Gender.MALE == Gender.MALE);  
   System.out.println(Gender.MALE.equals(Gender.MALE));

**Answer:**

true

true