

Enum for SCJP

1. Select all the correct statements regarding Enums (Select any 2 options)

- a) All enums are subclasses of interface `java.lang.Enum`.
- b) Enums is simply a data structure and hence it is not compiled to a .class file.
- c) Enums enable you to define a new data type. For example, If you create an Enum 'Days' you can declare a variable of type 'Days'.
- d) All instances of Enums are serializable by default.

Answer:

c, d is correct;

- a) `java.lang.Enum` is a class and not an interface.
- b) Enums are compiled to a `.class` file.
- c) Enums enable you to define a new data type, in a manner similar to the way a class allows you to define a new data type.
- d) Enums are subclasses of class `java.lang.Enum`, which implements the interface `java.io.Serializable`. If a class implements the interface `java.io.Serializable`, its objects are serializable.

2. What is the output of the following code:

```
line1> public enum IceCream {
line2>     VANILLA ("white"),
line3>     STRAWBERRY ("pink"),
line4>     WALNUT ("brown"),
line5>     CHOCOLATE ("dark brown");
line6>
line7>     String color;
line8>
line9>     IceCream (String color) {
line10>         this.color = color;
line11>     }
line12>
line13>     public static void main (String[] args) {
line14>         System.out.println (VANILLA);
line15>         System.out.println (CHOCOLATE);
line16>     }
line17> }
```

Option:

- a) Compilation error : Cannot run an enum as a standalone application.
- b) Compilation error at line no 14 & 15 : Cannot access VANILLA and CHOLOCLATE in 'static' main method.
- c) No errors. Output:
VANILLA
CHOCOLATE
- d) No errors. Output:
white
dark brown

Answer :

c is correct;

- a) An enum can define a main method and it can be executed as a standalone application.
- b) The instances of enums defined in an enum are static and hence available to any static method defined within the enum.
- d) The overridden toString method in the Enum class returns the name of this enum constant, as contained in the declaration. You can override the toString method to return a customised String value.

3. What is the output of the following code:

```
line1> public enum Day {
line2>     MONDAY (1),
line3>     TUESDAY (2),
line4>     WEDNESDAY (3) {public String toString() {
                                return "Good Morning"; } },
line5>     THURSDAY (4),
line6>     FRIDAY (5),
line7>     SATURDAY (6),
line8>     SUNDAY (7);
line9>
line10>     int dayNumber;
line11>
line12>     Day (int dayNumber) {
line13>         this.dayNumber = dayNumber;
line14>     }
line15>
line16>     public static void main (String[] args) {
line17>         for (Day d : Day.values())
line18>             System.out.println (d);
line19>     }
line20> }
```

a) Compilation error at line number 4 : Invalid code.

b) Executes OK giving OUTPUT:

```
MONDAY
TUESDAY
Good Morning
THURSDAY
FRIDAY
SATURDAY
SUNDAY
```

c) Runtime error : Cannot execute enum 'Day' as a standalone application.

d) Executes OK giving OUTPUT:

```
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
```

Answer :

b is correct;

- a) The code is valid. A single instance of enum can override the methods available to it.
- c) An enum can define a main method and it can be executed as a standalone application.
- d) The toString method for enum instance 'WEDNESDAY' returns 'Good Morning'.

4. Examine the following code and select the correct options:

```
enum Rating {  
    AVERAGE,  
    GOOD,  
    EXCELLENT;  
  
    abstract String performance();  
}  
  
class Test {  
    public static void main (String[] args) {  
        System.out.println (Rating.AVERAGE);  
    }  
}
```

- a) What rubbish, An Enum can never define an abstract method.
- b) class Test cannot access the enum constant AVERAGE using the code "Rating.AVERAGE".
- c) The above mentioned code will fail to compile.
- d) If the enum 'Ratings' is defined as an abstract 'enum', it will compile successfully.

Answer :

c is correct; The code will fail to compile because the abstract method 'performance' needs to be implemented by all the enum constants, i.e., AVERAGE, GOOD and EXCELLENT.

a) It is OK to define an abstract method in an enum.

b) Since Enum is not defined in class Test, its constant 'AVERAGE' should be prefixed with the Enum name 'Ratings' to access it in class Test.

d) We cannot define abstract enums.

5. Examine the following code and select the correct options that follow:

```
public enum IceCream {
    VANILLA ("white"),
    STRAWBERRY ("pink"),
    WALNUT ("brown"),
    CHOCOLATE ("dark brown");

    String color;

    IceCream (String color) {
        this.color = color;
    }
}

public class Test {
    public static void main (String[] args) {
        IceCream vanilla = new IceCream ("white");
        System.out.println ( vanilla );
    }
}
```

- a) The code prints out 'white'.
- b) The code prints out 'VANILLA'.
- c) The code fails at runtime.
- d) Enums may not be instantiated.

Answer :

d is correct; Enums cannot be instantiated. The code mentioned in this question will not compile.

6. Select all the incorrect options: (choose all that apply)

- a) Since enums are comparable to traditional classes, they may not be arguments in switch statements.
- b) Enums may not extend another class/ enum.
- c) Enums inherit the `java.lang.Object` class.
- d) Enums may be extended by another Enum.

Answer :

a, d is correct;

b) Enums can never extend any other class/ Enum.

c) All the Enums are subclasses of java.lang.Enum, which inherits java.lang.Object class.

7. Examine the following code: (choose all that apply)

```
enum Size {
    XS ("Extra Small") { int age() { return 5; }},
    S ("Small") { int age() { return 8; }},
    M ("Medium") { int age() { return 10; }},
    L ("Large") { int age() { return 15; }},
    XL ("Extra Large") { int age() { return 20; }};

    String description;

    Size (String desc) {
        this.description = desc;
    }
}

> // INSERT CODE HERE
}

class Test {
    public static void main (String[] args) {
        for (Size size : Size.values())
            System.out.println (size);
    }
}
```

Select the correct lines of code, which when inserted at line number 14, will print out the following output:

```
XS
S
M
L
XL
```

Options:

- a) `int age() { return 0; }`
- b) `abstract int age();`
- c) `short age() { return 100; }`
- d) `abstract short age();`

Answer:

- **a, b is correct;** Since all the enum constants, i.e., XS, S, M, L, XL implement the age method, this method can either be defined as a non-abstract method, or as an abstract method.
- Options c and d are not valid because the method "short age()" cannot override the method "int age()".
 - c) The method "short age() { return 100; }" cannot override either of the methods "int age()" and "abstract int age()".
 - d) The method "abstract short age() { return 100; }" cannot override either of the methods "int age()" and "abstract int age()".

8. Examine the following code:

```
import java.util.*;

enum Colours {
    YELLOW (Personality.EXPRESSIVE),
    GREEN (Personality.AMIABLE),
    RED (Personality.ASSERTIVE),
    BLUE (Personality.ANALYTICAL);

    Personality personality;

    Colours (Personality personality) {
        this.personality = personality;
    }

    enum Personality {ASSERTIVE, EXPRESSIVE, AMIABLE, ANALYTICAL };
}

class TestColours {
    public static void main (String[] args) {
        // INSERT LINE OF CODE HERE
    }
}
```

Which line of code when replaced with "// INSERT LINE OF CODE HERE", will output a value "true":

- a) System.out.println (Colours.Personality.ASSERTIVE instanceof Colours.Personality);
- b) System.out.println (Personality.EXPRESSIVE instanceof Personality);
- c) System.out.println (EXPRESSIVE instanceof Personality);
- d) None of the above

Answer :

a is correct; Enums may be used as a operand for the instanceof operator in the same way that it can be used by a class.

9. Examine the following code and select the correct options: (choose all that apply)

```
enum Rating {
    POOR (0.0, 5.0),
    AVERAGE (5.1, 7.0),
    GOOD (7.0, 8.5),
    EXCELLENT (8.6, 9.9);

    double lowerLimit, upperLimit;

    Rating (double ll, double ul) {
        this.lowerLimit = ll;
        this.upperLimit = ul;
    }

    int raise() {
        switch (this) {
            case POOR :      return 0;
            case AVERAGE :  return 5;
            case GOOD       :    return 20;
            case EXCELLENT:  return 45;
        }
        return 0;
    }
}

class Appraisal {
    public static void main (String args[])
        throws NumberFormatException {
        double currentSalary = 100;
        double increment = 0;

        for (Rating r : Rating.values()) {
            increment = currentSalary/ 100 * r.raise();
            System.out.println
                (r + " Performance,      Revised Salary = "
                 + (currentSalary + increment));
        }
    }
}
```

Options:

a) The above code will not compile: enums cannot be used as an argument to switch statement.

b) Code will compile, giving the following output:

```
POOR Performance, Revised Salary = 100.0
AVERAGE Performance, Revised Salary = 105.0
GOOD Performance, Revised Salary = 120.0
EXCELLENT Performance, Revised Salary = 145.0
```

c) The code will compile, giving the same results as mentioned in option 'b', if the enum Rating is defined as

```
enum Rating {
    POOR (0.0, 5.0) { int raise() { return 0; } },
    AVERAGE (5.1, 7.0) { int raise() { return 5; } },
    GOOD (7.0, 8.5) { int raise() { return 20; } },
    EXCELLENT (8.6, 9.9) { int raise() { return 45; } };

    double lowerLimit, upperLimit;

    Rating (double ll, double ul) {
        this.lowerLimit = ll;
        this.upperLimit = ul;
    }

    abstract int raise();
}
```

d) The constructor of an enum cannot accept 2 method parameters.

Answers :

b, c is correct;

- a) Enums can be used in switch statements.
- c) An enum constant can override any number of methods available to it.
- d) An enum can define more than 1 constructor (similar to a class) and its constructor can accept more than 1 method parameter.

10. Examine the following code and select the correct options:

```
enum Shape {
    CIRCLE (0, "red"),
    TRIANGLE (3),
    SQUARE (4),
    RECTANGLE (4),
    PENTAGON (5),
    HEXAGON (6, "yellow"),
    OCTAGON (8, "pink");

    int numberOfSides;
    String shapeColor;

    Shape (int sides) { numberOfSides = sides; }
    Shape (int sides, String colour) {
        numberOfSides = sides;
        shapeColor = colour;
    }

    public static void main (String[] args) {
        for (Shape s : Shape.values())
            System.out.println(s);
    }
}
```

Options:

a) Output of the code is as follows:

```
CIRCLE
TRIANGLE
SQUARE
RECTANGLE
PENTAGON
HEXAGON
OCTAGON
```

b) Output of the code is as follows:

```
CIRCLE red
TRIANGLE
SQUARE
RECTANGLE
PENTAGON
HEXAGON yellow
OCTAGON pink
```

c) The code will not compile : Cannot define more than one constructor in an enum.

d) The code will compile successfully, but throw a runtime error.

Answer :

a is correct;

- b) The default implementation of the toString method returns the name of the enum constant. To get any other customised value, override the toString method.
- c) The code will compile. An enum can define more than 1 constructor.
- d) There is no runtime error with the code.

11. Examine the following code:

```
public enum Day {
    MONDAY (1),
    TUESDAY (2),
    WEDNESDAY (3),
    THURSDAY (4),
    FRIDAY (5),
    SATURDAY (6),
    SUNDAY (7);

    int dayNumber;

    Day (int dayNumber) {
        this.dayNumber = dayNumber;
    }

    // INSERT CODE HERE

    public static void main (String[] args) {
        for (Day d : Day.values())
            System.out.println (d);
    }
}
```

Select from the following options the correct method, which when inserted in the above enum line, will print out the following result:

- 1 day of week
- 2 day of week
- 3 day of week
- 4 day of week
- 5 day of week
- 6 day of week
- 7 day of week

Options:

- a)

```
public String toString () {
    return dayNumber + " day of week";
}
```

- b)

```
public String name () {
    return dayNumber + " day of week";
}
```

Answer :

a is correct; name is a final method, which cannot be overridden. It returns the name of the enum constant, exactly as declared in its enum declaration.

12. Examine the following code:

```
1. enum Animals {
2.     DOG("woof"), CAT("meow"), FISH("burble");
3.     String sound;
4.     Animals(String s) { sound = s; }
5. }
6. class TestEnum {
7.     static Animals a;
8.     public static void main(String [] args) {
9.         System.out.println(a.DOG.sound + " " + a.FISH.sound);
10.    }
11. }
```

What is the result?

- A. woof burble
- B. Multiple compilation errors
- C. Compilation fails due to an error on line 2
- D. Compilation fails due to an error on line 3
- E. Compilation fails due to an error on line 4
- F. Compilation fails due to an error on line 9

Answer:

- A is correct; enums can have constructors and variables.
- B, C, D, E, and F are incorrect; these lines all use correct syntax. (Objective 1.3)

13. Examine the following code:

```
4. class Announce {
5.     public static void main(String[] args) {
6.         for(int __x = 0; __x < 3; __x++) ;
7.         int #lb = 7;
8.         long [] x [5];
9.         Boolean []ba[];
10.        enum Traffic { RED, YELLOW, GREEN };
11.    }
12. }
```

What is the result? (Choose all that apply.)

- A. Compilation succeeds
- B. Compilation fails with an error on line 6
- C. Compilation fails with an error on line 7
- D. Compilation fails with an error on line 8
- E. Compilation fails with an error on line 9
- F. Compilation fails with an error on line 10

Answer:

- C, D, and F are correct. Variable names cannot begin with a #, an array declaration can't include a size without an instantiation, and **enums can't be declared within a method.**
- A, B, and E are incorrect based on the above information. (Objective 1.3)

14. Examine the following code:

```
3. public class TestDays {
4.     public enum Days { MON, TUE, WED };
5.     public static void main(String[] args) {
6.         for(Days d : Days.values() )
7.             ;
8.         Days [] d2 = Days.values();
9.         System.out.println(d2[2]);
10.    }
11. }
```

What is the result? (Choose all that apply.)

- A. TUE
- B. WED
- C. The output is unpredictable
- D. Compilation fails due to an error on line 4
- E. Compilation fails due to an error on line 6
- F. Compilation fails due to an error on line 8
- G. Compilation fails due to an error on line 9

Answer:

- B** is correct. Every enum comes with a `static values()` method that returns an array of the enum's values, in the order in which they are declared in the enum.
- A, C, D, E, F, and G** are incorrect based on the above information. (Objective 1.3)