Selection 2

Operation	Example
`if`	`if (x > 0) { sout("Positive"); }`
`if ... else`	`if (x > 0) { sout("Positive"); } else { sout("Non-positive"); }`
`if ... elseif ... else`	`if (x > 0) { sout("Positive"); } else if (x < 0) { sout("Negative"); } else { sout("Zero"); }`
`switch`	`switch (x) { case 1: sout("One"); break; case 2: sout("Two"); break; default: sout("Other"); }`
Nested `if` statement	`if (x > 0) { if (x % 2 == 0) { sout("Positive and Even"); } }`
Conditional Operators	`boolean isTrue = (x > 0) && (x < 10);`

Operation	Description	Example	Output Example
`+` (Binary addition)	Binary arithmetic addition.	`int a = 5 + 3;`	`a` equals 8
`-` (Binary subtraction)	Binary arithmetic subtraction.	`int b = 8 - 4;`	`b` equals 4
`*`(Multiplication)	Binary multiplication.	`int c = 6 * 4;`	`c` equals 24
`/`(Division)	Binary division.	`int d = 20 / 5;`	`d` equals 4
`%`(Modulus)	Binary modulus (remainder of division).	`int e = 17 % 5;`	`e` equals 2
`+`(Unary plus)	Unary plus.	`int f = +5;`	`f` equals 5
`-`(Unary minus)	Unary minus.	`int g = -7;`	`g` equals -7
`++var`(Pre-increment)	Pre-increment.	`int h = 5; ++h;`	`h` becomes 6
`--var`(Pre-decrement)	Pre-decrement.	`int i = 10; --i;`	`i` becomes 9
`var++`(Post-increment)	Post-increment.	`int j = 5; j++;`	`j` becomes 6
`var--`(Post-decrement)	Post-decrement.	`int k = 10; k--;`	`k` becomes 9
`!`(Not)	Logical NOT.	`boolean l = !flag;`	`l` becomes `false`
`<`(Less than)	Less than comparison.	`boolean m = (5 < 10);`	`m` becomes `true`
`>`(Greater than)	Greater than comparison.	`boolean n = (10 > 5);`	`n` becomes `true`
`!=`(Not equal to)	Not equal to comparison.	`boolean o = (3 != 5);`	`o` becomes `true`
`==`(Equal to)	Equal to comparison.	`boolean p = (5 == 5);`	`p` becomes `true`

Operation	Description	Example	Output Example
`<=`(Less than or equal to)	Less than or equal to comparison.	`boolean q = (10 <= 15);`	`q` becomes `true`
`>=`(Greater than or equal to)	Greater than or equal to comparison.	`boolean r = (20 >= 10);`	`r` becomes `true`
`&&`(Logical AND)	Conditional AND.	`boolean s = (true && false);`	`s` becomes `false`
`||`(Logical OR)	Conditional OR.	`boolean t = (true || false);`	`t` becomes `true`
`^`(Logical XOR)	Conditional XOR (exclusive OR).	`boolean u = (true ^ false);`	`u` becomes `true`
`=`(Assignment)	Assignment.	`int w = 5;`	`w` becomes `5`
`+=`(Add and assign)	Add and assign.	`int x = 5; x += 3;`	`x` becomes `8`
`-=`(Subtract and assign)	Subtract and assign.	`int y = 10; y -= 4;`	`y` becomes `6`
`*=`(Multiply and assign)	Multiply and assign.	`int z = 20; z *= 2;`	`z` becomes `40`
`/=`(Divide and assign)	Divide and assign.	`int m = 15; m /= 3;`	`m` becomes `5`
`%=`(Modulus and assign)	Modulus and assign.	`int n = 12; n %= 5;`	`n` becomes `2`

1. Fill the blanks so the code prints `Path A`.

```java
class punchcard {
    public static void main(String[] args) {
        int x = 4;
        ______ (x) {
            ____ 4:
                System.out.println("Path A");
                _____;
            default:
                System.out.println("Path B");
                break;
        }
    }
}
```

A) `switch`, `case`, `break`
B) `if`, `else`, `continue`
C) `if`, `else`, `break`
D) `switch`, `case`, `continue`

---

2. Given the following code snippet, what will be the output?

```java
class punchcard {
    public static void main(String[] args) {
        int y = 4;
        switch (y) {
            default:
                System.out.println("Default");
            case 1:
                System.out.println("One");
                break;
            case 2:
                System.out.println("Two");
            case 3:
                System.out.println("Three");
                break;
        }
    }
}
```

A) `Default`

B) `One`

C) `Default \n One`

D) `Default \n One \n Two \n Three`

E) `Default \n Two \n Three`

---

3. What is the output of the following Java code?

```
class punchcard {
    public static void main(String[] args) {
        int z = 2;
        switch (z) {
            case 1:
                System.out.println("Case 1");
                break;
            case 2:
            case 3:
                System.out.println("Case 2 or 3");
                break;
            default:
                System.out.println("Default");
        }

    }
}
```

A) `Case 1`

B) `Case 2 or 3`

C) `Default`

D) `Case 2 or 3 \n Default`

E) `No output`

F) `Error`

---

4. What is the output of the following Java code?

```
class punchcard {
    public static void main(String[] args) {
        int input = 7;
        switch (input % 5) {
            case 1:
                System.out.println("One");
                break;
            case 2:
                System.out.println("Two");
                break;
            case 3:
                System.out.println("Three");
                break;
            default:
                System.out.println("Default");
        }
    }
}
```

A) One
B) Two
C) Three
D) Default

- Question 5 to 7 are extra. You can skip them if you want.
- These switch features require Java 14 or higher.

5. What is the output of the following Java code?

```
class punchcard {
    public static void main(String[] args) {
        int dayofweek = 6;
        switch (dayofweek) {
            case 1, 2, 3, 4, 5:
                System.out.println("wait, we have class today?!!");
                break;
            case 6, 7:
                System.out.println("going to a partttttttttttty");
                break;
            default:
                System.out.println("not a valid day!");
        }
    }
}
```

A) wait, we have class today?!!
B) going to a partttttttttttty
C) Not a valid day!

D) Error

---

6. What is the output of the following Java code?

```
[class](class) punchcard {
    public static void main(String[] args) {
        int dayOfWeek = 6;
        switch (dayOfWeek) {
            case 1, 2, 3, 4, 5 -> System.out.println("wait, we have class today?!!");
            case 6, 7 -> System.out.println("going to a partttttttttttty");
            default -> System.out.println("Not a valid day!");
        }
    }
}
```

A) wait, we have class today?!!
B) going to a partttttttttttty
C) Not a valid day!
D) Error

---

7. What is the output of the following Java code?

```
class punchcard {
    public static void main(String[] args) {
        int dayOfWeek = 7;
        int day = 23, month = 5, year = 2021;

        String date = switch (dayOfWeek) {
            case 1 -> "Monday";
            case 2 -> "Tuesday";
            case 3 -> "Wednesday";
            case 4 -> "Thursday";
            case 5 -> "Friday";
            case 6 -> "Saturday";
            case 7 -> "Sunday";
                default -> "Invalid day of week!";
        }
        + ", " + day + ". "
            + switch (month) {
                case 1 -> "January";
                case 2 -> "February";
                case 3 -> "March";
                case 4 -> "April";
                case 5 -> "May";
                case 6 -> "June";
                case 7 -> "July";
                case 8 -> "August";
                case 9 -> "September";
                case 10 -> "October";
                case 11 -> "November";
                case 12 -> "December";
                default -> "Invalid month!";
            }
        + " " + year;
        System.out.println(date + "\n");
    }
}
```

A) Sunday, 23. May 2021
B) Invalid day of week!, 23. Invalid month! 2021
C) Invalid day of week!, 23. May 2021
D) Thursday, 23. April 2021
E) Error

---

8. You are creating a program to simulate a color-matching game where each color corresponds to a specific number. Your task is to write a code that takes a user input of type string (`color`). The input color is matched with a numeric code based on the following color chart:

The color codes are as follows: - Red - `101` - Blue - `202` - Green - `303` - Yellow - `404` - Orange - `505`

Then print the corresponding numeric code to the console.

Example function signature:

```
public class Main {
    public static void main(String[] args) {
        // make a scanner object

        // get the color from the user

        // use if-else or switch-case to print the corresponding numeric code
    }
}
```

Constraints:

- The input `color` will always be a valid string among: “Red”, “Blue”, “Green”, “Yellow”, “Orange”.

Example:

This question challenges candidates to use a `switch-case` statement to match the input color string with its corresponding numeric code efficiently and creatively. The goal is to implement a compact and elegant `getColorCode` function using the switch-case construct.

Answers:

1. A
2. C
3. B
4. B
5. B
6. B
7. A
8. a possible solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String color = scanner.nextLine();
        switch (color) {
            case "Red":
                System.out.println("101");
                break;
            case "Blue":
                System.out.println("202");
                break;
            case "Green":
                System.out.println("303");
                break;
            case "Yellow":
                System.out.println("404");
                break;
            case "Orange":
                System.out.println("505");
                break;
            default:
                System.out.println("Invalid color");
        }
    }
}
```