1.B

A **switch statement** allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

The following rules apply to a **switch** statement −

- The variable used in a switch statement can only be integers, convertable integers (byte, short, char), strings and enums.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break* statement is reached.
- When a *break* statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A *switch* statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

2.D

```
public class Q2 {
	public static void main (String[] args)
	{
		int meal = 5 ;
		int tip =2 ;
		int total = meal + (meal>6 ? ++tip : --tip);
		// meal'a yani 5'e --tip yani tip'in 1 eksiği eklenecek
		// 5 + 1 =6
		System.out.println(total);
	}
}
```

3.C

```
public class Q3 {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		String john = "john";
		String jon = new String(john);
		System.out.print((john == jon) + " " + (john.equals(jon)));
	}
}
```

4.B

```
public class Q4 {
	public static void main(String[] args) {
		int plan = 1;
		plan = plan++ + --plan;
		if (plan == 1) {
			System.out.print("Plan A");
		} else {
			if (plan == 2)
```

```
                                System.out.print("Plan B");
                        else
                                System.out.print("Plan C");
                }

        }
}
```

5.C

Default statement bir değer almaz diğer case lerdeki durumların hiç birisine uymuyorsa default çalışır.

6.B

```
public class Q6 {
        public static void main(String[] args) {
                long thatNumber = 5 >= 5 ? 1 + 2 : 1 * 1;
                if (++thatNumber < 4)
                        thatNumber += 1;
                System.out.print(thatNumber);
        }
}
```

7.B

8.C

ternary expression if-else yapısının kısayolu gibidir.

9.C

```
public class Q9 {
        public void calculateResult(Integer candidateA, Integer candidateB) {
                boolean process = candidateA == null || candidateA.intValue() < 10;
                boolean value = candidateA && candidateB;
                //The operator && is undefined for the argument type(s) java.lang.Integer,
                System.out.println(process || value);
        }

        public static void main(String[] unused) {
                new Q9().calculateResult(null, 203);

        }
}
```

10.A

```
public class Q10 {
        public final static void main(String[] args) {
                int pterodactly = 6;
                long triceratops = 3;
                if (pterodactly % 3 >= 1) //6%3 0 oalcağı için if içine girmez
                        triceratops++;
                triceratops--;

                System.out.println(triceratops);
        }
}
```

11.D ??

12.D

```
public class Q12 {
```

```
public static void main(String[] args) {
	int flair =15;
	if(flair >=15 && flair <37 ) {
		System.out.println("not enough"); // f:15 olduğu için buraya girecek
	} if(flair==37) {
		System.out.println("just right");
	} else { //f:37 olmadığı için buraya girecek
		System.out.println("too many");
	}
}
}
// hem not enough hem de too many tazdırılacak
```

13.C ??

14.D

Tabloda X ve Y nin boolean olasılıklaı değerlendirilmiş. && ifadesiyle kesişimini alabiliriz.

15.C

```
public class Q15 {
	public static void main(String[] args) {
		int hops = 0;
		int jumps = 0;
		jumps = hops++;
		if (jumps) // if içerisi boolean olmak zorundadır.
			System.out.println("jump");
		else
			System.out.println(hop);
	}

}
```

16.B

++v değeri artırır ve artırılmış değeri dönerken, v—değeri arkaplanda azaltır ama orijinal değerini döner

17.B

```
public class Q17 {
	public static void main(String[] args) {
		int tiger = 2;
		short lion = 3;
		long winner = lion + 2 * (tiger + lion);
		// parantez icinin onceligi vardir
		System.out.println(winner);
	}
}
```

18.B

Long int'e dönüştürülemeyeceği için hata verecektir.

19.D

Kod derlenmez çünkü day ifadesinin boolean olması gerekir.

20.B

```
public class Q20 {
	public static void main(String[] args) {
		int leaders = 10 * (2 + (1 + 2 / 5));
		int followers = leaders * 2;
		System.out.println(leaders + followers < 10 ? "too few" : "too many");

	}

}
```

21.B

```
public class Q21 {
public static void main(String[] args) {
	System.out.println(5+6+"7"+8+9);
	//ilk başta iki int değeri topladı 11
	// işin içine string girince artık stringe geçti
	//sayıları string olarak yan yana yazmaya başladı
}
}
```

22.B

23.B

```
public class Q23 {
	public static void main(String[] args) {
		int dog = 11;
		int cat = 3;
		int partA = dog / cat;
		int partB = dog % cat;
		int newDog = partB + partA * cat;
		System.out.println(newDog);
	}
}
```

24.B

```
public class Q24 {
	public static void main(String[] args) {
		int flavors = 30;
		int eaten = 0;
		switch (flavors) {
		case 30: //break konulmadığı için sırayla hepsine girecek
			eaten++; // eaten 1 oldu
		case 40:
			eaten += 2; // eaten 3 oldu
		default:
			eaten--; //eaten 2 oldu
		}
		System.out.println(eaten);
	}
}
```

25.C

26.A

(==) ve Equals() methodları'nın ikisi de farklı 2 değeri karşılaştırmak için kullanılır. (==) operator'ü 2 nesneyi karşılaştırırken, Equals() methodu nesnenin içerdiği string'i karşılaştırır. Yani kısaca (==) operatörü 2 nesnenin referans değerlerini karşılaştırırken Equals() methodu sadece içeriği karşılaştırır.

27.B

myTestVariable null değilse myTestVariable.equals(null) ifadesi false dönecektir.

28.D

```
 else if (streets && intersections > 1000)
```

satırında street degeri int olduğu halde boolean gibi işlem yapılmaya calısılmıs.

29.C

```
&&       Conditional-AND
&        Bitwise AND
```

30.C

```
public class Q30 {
	public static void main(String[] args) {
		boolean w = true, z = false;
		int x = 10, y = 5;
		x = w ? y++ : y--;
		w = !z;
		System.out.println((x + y) + "" + (w ? 5 : 10));

	}
}
```

31.A

```
public class Q31 {
	public static void main(String[] args) {
		String bob = new String("bob");
		String notBob = bob;
		System.out.print((bob == notBob) + " " + (bob.equals(notBob)));

	}
}
```

32.B

33.B

False^true = false

False^false = false

34.A ?

```
public class Q34 {
	public static void main(String[] data) {
		if (data.length >= 1
				&& (data[0].equals("sound")
						|| data[0].equals("logic"))
				&& data.length < 2) {
			System.out.print(data[0]);
		}
	}
}
```

35.C

https://introcs.cs.princeton.edu/java/11precedence/

Level	Operator	Description	Associativity

16	`[]`
	`.`
	`()`
		access array element
		access object member
		parentheses
			left to right

15	`++` `--`	unary post-increment unary post-decrement	not associative

14	`++`
	`--`
	`+`
	`-`
	`!`
	`~`
		unary pre-increment
		unary pre-decrement
		unary plus
		unary minus
		unary logical NOT
		unary bitwise NOT
			right to left

13	`()`
	`new`
		cast
		object creation
			right to left

12	`* / %`	multiplicative	left to right

11	`+ -`
	`+`
		additive
		string concatenation
			left to right

10	`<< >>`
	`>>>`
		shift	left to right

9	`< <=`
	`> >=`
	`instanceof`
		relational	not associative

8	`==`
	`!=`
		equality	left to right

7	`&`	bitwise AND	left to right
6	`^`	bitwise XOR	left to right
5	`|`	bitwise OR	left to right
4	`&&`	logical AND	left to right

3	`||`	logical OR	left to right
2	`?:`	ternary	right to left
1	`=` `+=` `-=` `*=` `/=` `%=` `&=` `^=` `|=` `<<=` `>>=` `>>>=`	assignment	right to left

36.C

Mantıksalişlemlerde ve operatorü varsa sonucun true olması için tüm bileşenler true olmalıdır.

37.C

38.D

```
case expressions must be constant expressions
```

39.B

40.B

```
public class Q40 {
        public static void main(String[] argumants) {
        int turtle = 10 * ( 2 + ( 3 + 2 ) / 5);
        int hare = turtle < 5 ? 10 : 25;
        System.out.print(turtle < hare ? "Hare wins!" : "Turtle Wins");
    }
}
```

41.A

```
public class Q41 {
        public static int getResult(int threshold){
        return threshold > 5 ? 1 : 0 ;
    }

    public static void main(String[] argumants) {
    System.out.println(getResult(5)+getResult(1)+getResult(0)+getResult(2)+"");
   // System.out.print(5+2+3+4+5+6+8+"");

    }
}
```

42.A

```
public class Q42 {
        public String runTest(boolean snipper, boolean roller) {
                if (snipper = roller)
                        return "up";
                else
                        return roller ? "down" : "middle";
        }

        public static void main(String[] args) {
                final Q42 tester = new Q42();
                System.out.println(tester.runTest(false, true));
}}
```

43.B

44.A

```
public class Q44 {
	public static void main(String[] args) {
		int characters = 5;
		int story = 3;
		double movieRating = characters <= 4 ? 3 : story > 1 ? 2 : 1;
		System.out.println(movieRating);
	}
}
```

45.B

Soru 1'de verilen özellikler incelenebilir.

46.B ?

```
public class Q46 {
	public static void main(String[] weather) {

	System.out.print(weather[0]!=null&&weather[0].equals("sunny")&& !false?"Go Outside" : "Stay 
Inside");

    }
}
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0 at com.company.Main.main(Main.java:10)

47.D

- The operator ! is undefined for the argument type(s) int

48.C

49.A

Sou 35'te verien tablo incelenebilir.

50.C

```
public class Q50 {
	public static String play(int toy, int age) {
		final String game;
		if (toy < 2)
			game = age > 1 ? 1 : 10; // p1
		// Type mismatch: cannot convert from int to String
		else
			game = age > 3 ? "Ball" : "Swim";// p2
		return game;
	}

	public static void main(String[] args) {
		System.out.print(play(5, 2));

	}
}
```