when statement in Kotlin

In Kotlin, switch statement of Java is replaced with `when` statement. Similar to the if-else statement, you can use `when` as an expression or as a statement. There are three ways in which you can use `when` statement-

- when statement with argument
- when statement without argument
- when as an expression

when statement with argument

when with argument behaves as a switch statement. The difference between when and switch is in the way we write syntax.

Syntax

```
when(argument){
   value1 -> {
      //code to be executed
   }
   value2 -> {
      //code to be executed
   }
   ...
   else -> {
      //code to be executed
   }
}
```

> **Note-** value1, value2, ..., valuen are called branch conditions. If you have only one statement to execute then no need to mention curly braces in the branch condition. You can think `else` as a switch statement's default label.

Example

```
public fun main(args: Array<String>) {
   print("Enter a character: ")
   val i = readLine()
   when(i){
       "a"-> print("You entered vowel a")
       "e"-> print("You entered vowel e")
       "i"-> print("You entered vowel i")
       "o"-> print("You entered vowel o")
       "u"-> print("You entered vowel u")
       else -> print("You entered consonant")
   }
}
```

Output

```
Enter a character: a
You entered vowel a
```

when statement without argument

If no argument is provided to `when` statement, then `when` works as if-else if statement. Here branch conditions must be specified as boolean expression.

Syntax

```
when {
   condition1 -> {
      //code to be executed
   }
   condition2 -> {
      //code to be executed
   }
   ...
   else -> {
      //code to be executed
   }
}
```

Example

```
public fun main(args: Array<String>) {
   val i = 12
   when{
       i > 0 -> print("Number is positive.")
       i == 0 -> print("Number is zero.")
       i < 0 -> print("Number is negative.")
   }
}
```

Output

```
Number is positive.
```

when as an expression

On using `when` as an expression, then it returns a value which you can store it in a variable.

Example

```
public fun main(args: Array<String>) {
   var i = 30
   var j = 40
   println("Choose any one")
   println("1. Addition")
   println("2. Subtraction")
   println("3. Multiplication")
   println("4. Division")
   print("Enter your choice: ")
   val choice = readLine()
   val result = when(choice) {
       "1" -> i + j
       "2" -> i - j
       "3" -> i * j
       "4" -> i / j
       else -> "Invalid choice entered"
   }
   print("Output: $result")
}
```

Output

```
Choose any one
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice: 1
Output: 70
```

Possible forms of when statement

1. You can combine more than one branch conditions with a comma.

```
public fun main(args: Array<String>) {
   val i: Int = 4
   when(i){
      1,2,3,4 -> print("Number is positive and less than 5")
      0 -> print("Number is zero")
      -1, -2 -> print("Number is negative and less than zero")
   }
}
```

Output

```
Number is positive and less than 5
```

2. You can check a value from a range or collection using in or !in keyword.

```
public fun main(args: Array<String>) {
   val i: Int = 85
   when(i){
      in 1..100 -> print("Number is between 1 and 100")
      !in 1..100 -> print("Number is less than zero or greater than 
100")
   }
}
```

Output

```
Number is between 1 and 100
```

3. Apart from constants, you can also use expression as a branch condition.

```
public fun main(args: Array<String>) {
   val i = "51"
   when(i){
      51.toString()-> print("You are lucky!")
      101.toString() -> print("You are champion")
   }
}
```

Output

```
You are lucky!
```

> **Note-** You are free to use constants, `in`, `!in` and expressions, all in the same `when` statement.