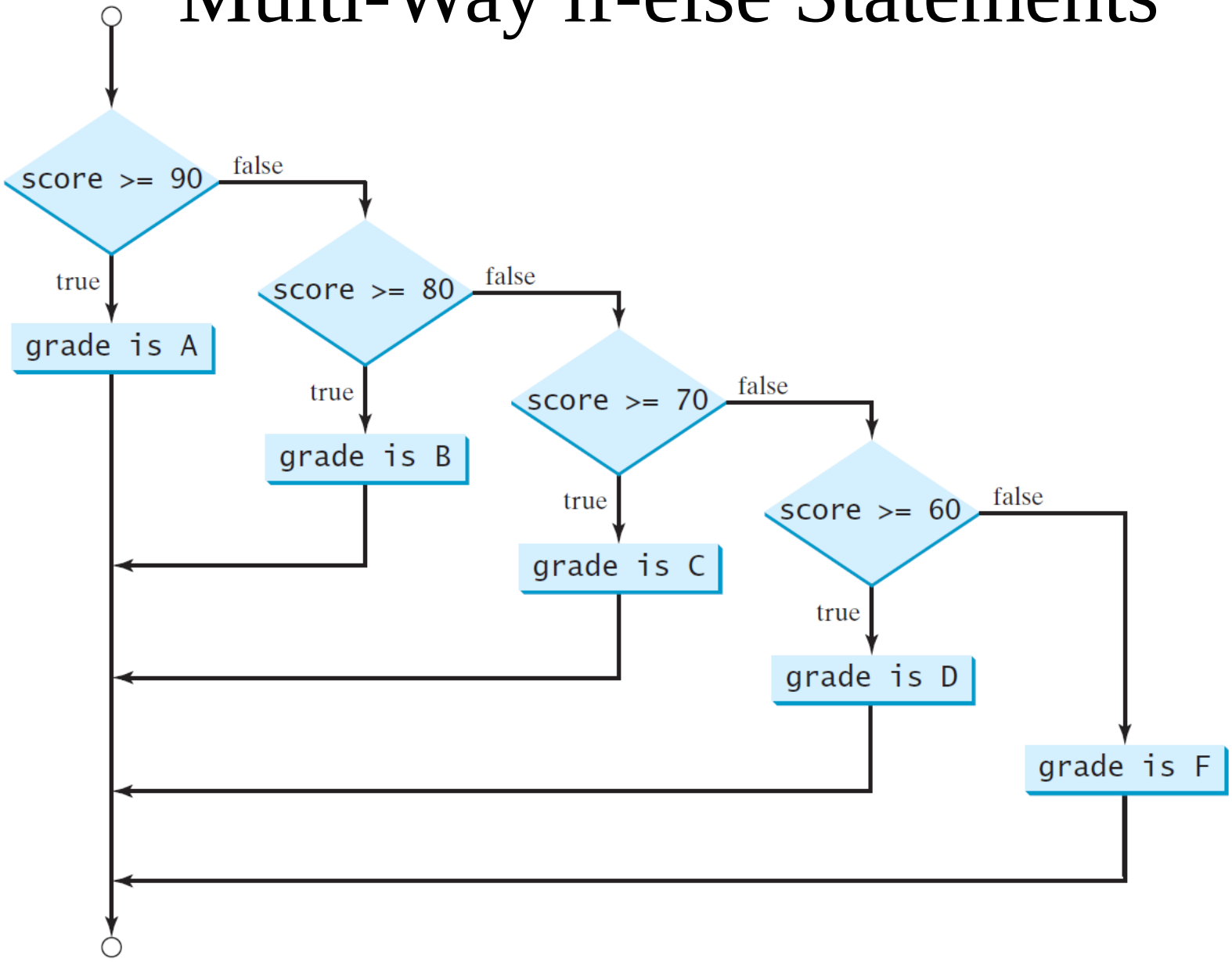
Multi-sided Selections
(if/elseif/else and switch)
Nested Selections

Multi-Way if-else Statements



Truth Table for Operator !

p	!p	Example (assume age = 24, weight = 140)
true	false	!(age > 18) is false, because (age > 18) is true.
false	true	!(weight == 150) is true, because (weight == 150) is false.

Conditional Expressions

```
if (x > 0)
  y = 1
else
  y = -1;
```

is equivalent to

```
y = (x > 0) ? 1 : -1;
```

(boolean-expression) ? expression1 : expression2

Given that a = 5, b = 10, and c = 50, what value goes into each of the variables declared below?

a) int result = (a > b) ? a : b;

b) String value = (c >= 100) ? “hello” : “goodbye”;

c) double number = (a + b <= c) ? (a + b) : c;

The Switch Statement

A common structure in many programming languages is the "case" statement.

This kind of statement is a lot like a multi-sided selection and in most cases, can be a lot more powerful and efficient than a multi-sided selection.

Java's version of a "case" statement is the `switch` statement.

The switch statement examines one value (usually, but not always a variable) and performs an action or actions based on that value.

The switch statement can only examine values that are castable into an int.

These include data of type int, byte, short, and char. You can't use a switch with any other data type.

The general format of a switch statement is:

```
switch (variable) {
    case value1:
       // execute if variable = value1
       break;
    case value2:
       // execute if variable = value2
       break;
    case value3:
       // execute if variable = value3
       break;
    default:
       // execute if variable is something else
}
```

You'll also notice the break; statement in each of the case blocks. These are optional, but most of the time you would want to keep them. To see why, run this program:

```
public class TrySwitch2 {
    public static void main(String[] args) {
        int number = 2;
        switch (number) {
            case 1:
                System.out.println("one");
            case 2:
                System.out.println("two");
            case 3:
                System.out.println("three");
            default:
                System.out.println("too many!");
        }
    }
}
```

- What output do you expect? What output do you get?
- This can actually come in handy:

```
public class TrySwitch2 {

    public static void main(String[] args) {

        int number = 5;
        switch (number) {
            case 1:
                    System.out.println("only one");
                break;
            case 2:
            case 3:
                System.out.println("a few");
                break;
            case 4:
            case 5:
            case 6:
                System.out.println("a handful");
                break;
            default:
                System.out.println("too many!");
        }
    }
}
```

//Try re-running this program with the values 1, 3, and 9 for the variable number.

Prior to Java 7, you could only use a switch statement with any data type that could be implicitly cast into an int (byte, short, char, and int). In the Java 7 release, the switch statement was given support for String objects:

```
public class FindProvCode {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.println("Enter full province/territory name:")
        String province = in.nextLine();
        String provAbbrev = "";

        switch (province.toLowerCase()) {
            case "alberta":
                provAbbrev = "AB";
                break;
            case "british columbia":
                provAbbrev = "BC";
                break;
            case "manitoba":
                provAbbrev = "MB";
                break;
            case "new brunswick":
                provAbbrev = "NB";
                break;
```

```
            case "newfoundland labrador":
                provAbbrev = "NL";
                break;
            case "nova scotia":
                provAbbrev = "NS";
                break;
            case "northwest territories":
                provAbbrev = "NT";
                break;
            case "nunavut":
                provAbbrev = "NU";
                break;
            case "ontario":
                provAbbrev = "ON";
                break;
            case "prince edward island":
            case "pei":
                provAbbrev = "PE";
                break;
            case "quebec":
                provAbbrev = "QC";
                break;
            case "saskatchewan":
                provAbbrev = "SK";
                break;
            case "yukon":
                provAbbrev = "YT";
                break;
        }
        System.out.println("Official abbreviation for " +
            province + ": " + provAbbrev);
    }
}
```

Rewrite the following selection statement as a switch statement:

```
import java.util.Scanner;

public class IfToSwitch {

    public static void main(String[] args) {

        Scanner keysIn = new Scanner(System.in);
        System.out.println("Enter the shipment status:");
        int status = keysIn.nextInt();

        String action = "";
        if (status == 1) {
            action = "Phone vendor and request delivery.";
        } else if (status == 2) {
            action = "Phone customer and warn of late shipment.";
        } else if (status == 3) {
            action = "Prepare shipping labels.";
        } else if (status == 4) {
            action = "Order additional product from vendor.";
        } else {
            action = "Error:  Invalid status code.";
        }
        System.out.println(action);
    }
}
```

```java
import java.util.Scanner;

public class Question1
{
    public static void main(String[] args)
    {
        Scanner keysIn = new Scanner(System.in);
        System.out.println("Enter the shipment status:");
        int status = keysIn.nextInt();
        String action = "";
        switch(status) {
            case 1:
                action = "Phone vendor and request delivery.";
                break;
            case 2:
                action = "Phone customer and warn of late shipment.";
                break;
            case 3:
                action = "Prepare shipping labels.";
                break;
            case 4:
                action = "Order additional product from vendor.";
                break;
            default:
                action = "Error:  Invalid status code.";
        }

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

Exercise 2:

Write a program using a switch statement that requests a number from 1 to 7.

Display the day of the week (e.g. Sunday, Monday..) corresponding to that number (1 is Sunday). Display an error message if the user enters anything other than a number from 1 to 7.

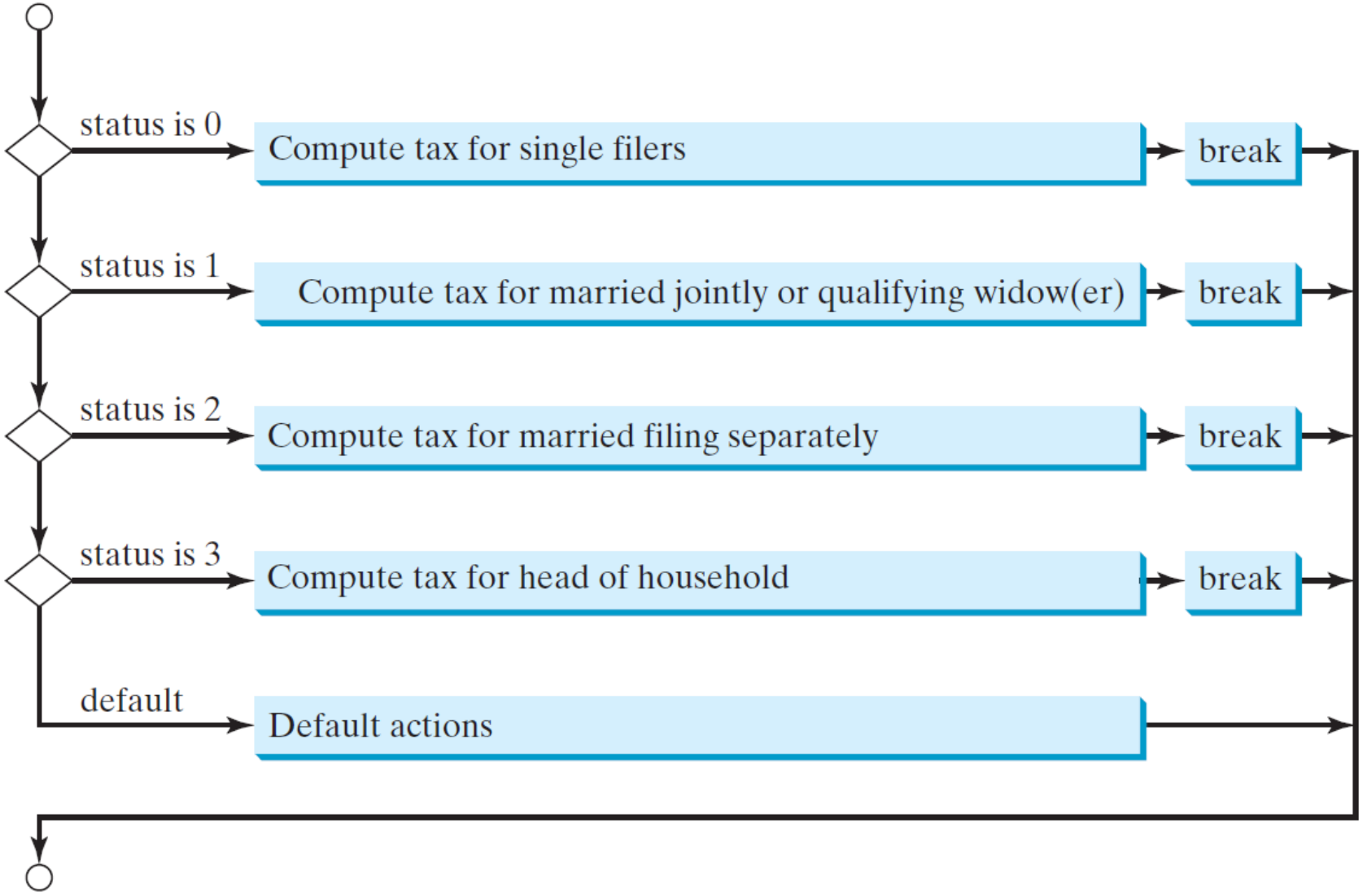
Pay close attention to the syntax of this statement:

- The variable you are testing must go after "switch" in brackets.
- Each case statement ends with a colon.
- Don't use braces if a case contains more than one statement.
- "default:" is optional and can be used when none of the cases apply (like an else in a multi-sided if)

`switch` Statements

```
switch (status) {
  case 0:  compute taxes for single filers;
           break;
  case 1:  compute taxes for married file jointly;
           break;
  case 2:  compute taxes for married file separately;
           break;
  case 3:  compute taxes for head of household;
           break;
  default: System.out.println("Errors: invalid status");
           System.exit(1);
}
```

`switch` Statement Flow Chart



`switch` Statement Rules

The switch-expression must yield a value of char, byte, short, or int type and must always be enclosed in parentheses.

The value1, ..., and valueN must have the same data type as the value of the switch-expression. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. Note that value1, ..., and valueN are constant expressions, meaning that they cannot contain variables in the expression, such as 1 + x.

```
switch (switch-expression) {
  case value1:  statement(s)1;
                break;
  case value2: statement(s)2;
                break;
  …
  case valueN: statement(s)N;
                break;
  default: statement(s)-for-default;
}
```

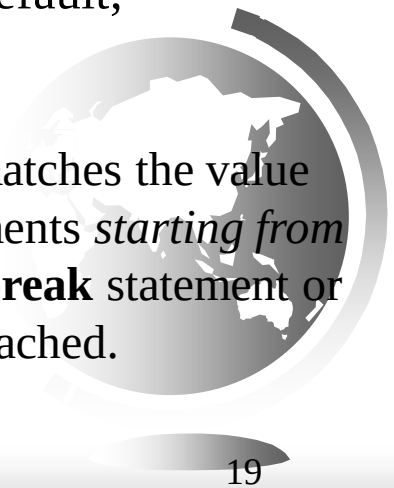
`switch` Statement Rules

The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

The default case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

```
switch (switch-expression) {
  case value1:  statement(s)1;
        break;
  case value2: statement(s)2;
        break;
  …
  case valueN: statement(s)N;
        break;
  default: statement(s)-for-default;
}
```

When the value in a **case** statement matches the value of the **switch-expression**, the statements *starting from this case* are executed until either a **break** statement or the end of the **switch** statement is reached.



animation

Trace switch statement

Suppose day is 2:

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

animation

Trace switch statement

Match case 2

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

animation

Trace switch statement

Fall through case 3

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

animation

Trace switch statement

Fall through case 4

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

animation

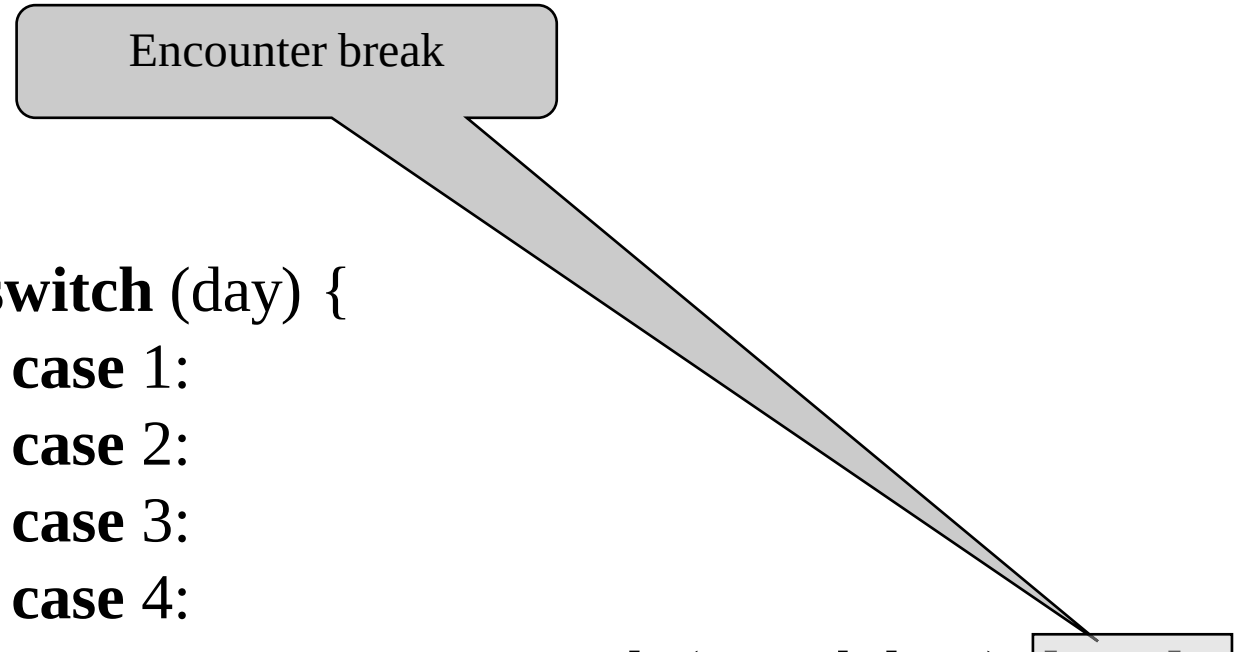
Trace switch statement

Fall through case 5

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

animation

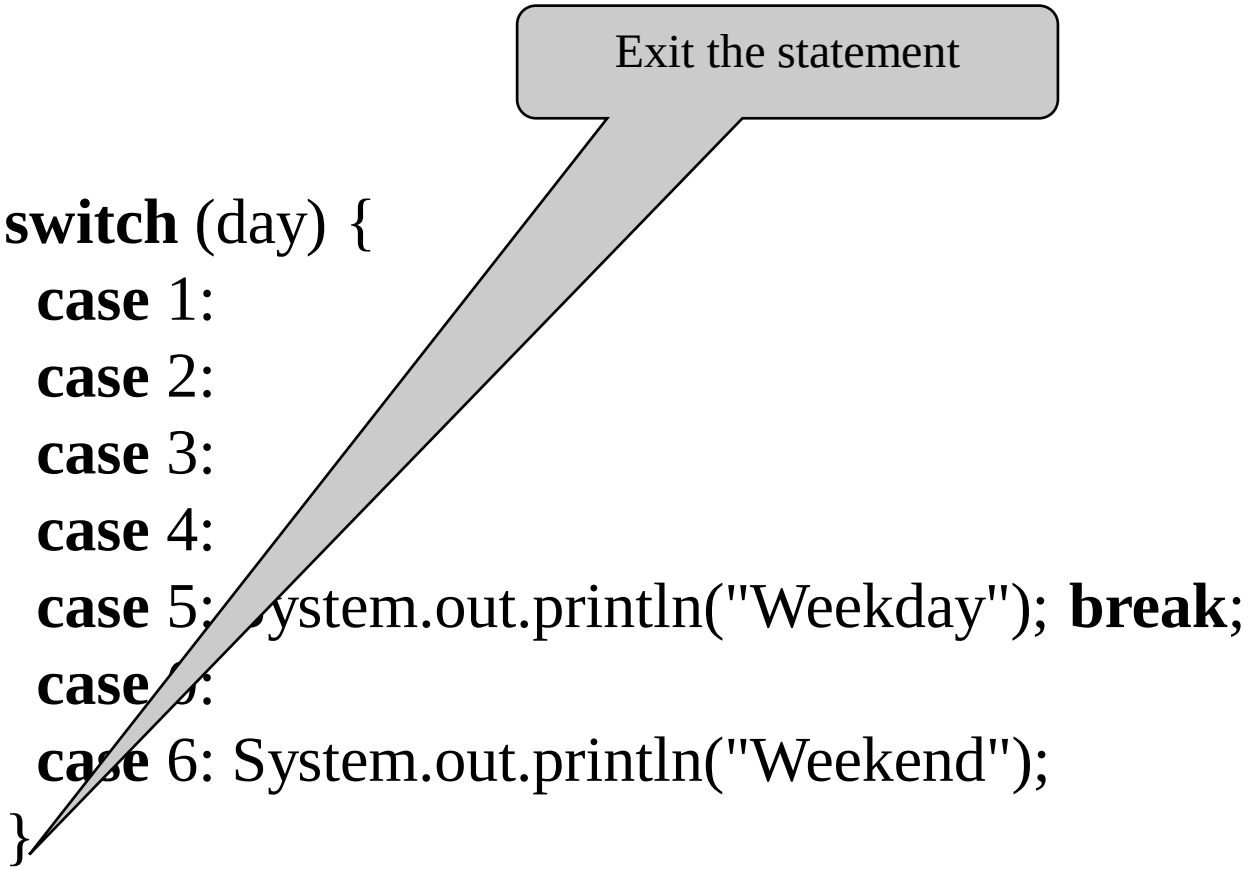
Trace switch statement




```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

animation

Trace switch statement




```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 6:
  case 6: System.out.println("Weekend");
}
```

Operator Precedence

- **var++, var--**
- **+, - (Unary plus and minus), ++var,--var**
- **(type) Casting**
- **! (Not)**
- ***, /, % (Multiplication, division, and remainder)**
- **+, - (Binary addition and subtraction)**
- **<, <=, >, >= (Relational operators)**
- **==, !=; (Equality)**
- **^ (Exclusive OR)**
- **&& (Conditional AND) Short-circuit AND**
- **|| (Conditional OR) Short-circuit OR**
- **=, +=, -=, *=, /=, %= (Assignment operator)**