Switch Statement

Jim Wilson
Mobile solutions developer & architect

@hedgehogjim jwhh.com

Overview

- Switch statement organization
- Branch control flow
- Branch ordering
- Allowable data types
- Choosing between switch and if-else

Switch statement

- Test value against multiple matches
- Transfer control based on match

```
switch (test-value) {
  case match-1:
    statements
    break;
  .
  .
  .
  case match-N:
    statements
    break;
  default:
    statements
}
```

Switch Statement

```
char sign = '-';
switch(sign)
  case '+':
    System.out.println("Positive");
    break;
  case '-':
    System.out.println("Negative");
    break;
  default:
    System.out.println("Sign not recognized");
    break;
}
System.out.println("Keep working...");
```

Switch Statement

```
char sign = '-';
switch(sign) {
  case '+':
    System.out.println("Positive");
    break;
  case '-':
    System.out.println("Negative");
    break;
  default:
    System.out.println("Sign not recognized");
    break;
}
System.out.println("Keep working...");
```

Branch Control Flow

```
String valueName = "two";
int total = 10;
switch(valueName) {
    case "one":
        total += 1;

    case "two":
        total += 2;

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

12

Branch Control Flow

```
String valueName = "one";
int total = 10;
switch(valueName) {
    case "one":
        total += 1;

    case "two":
        total += 2;

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

13

Branch Control Flow

```
String valueName = "one";
int total = 10;
switch(valueName) {
    case "one":
        total += 1;

    case "tw  :
        tota  += 2;

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

Branch Control Flow

```
String valueName = "one";
int total = 10;
switch(valueName) {
    case "one":
        total += 1;
        break;
    case "two":
        total += 2;
        break;
}
System.out.println(total);
```

11

BranchOrder.java

```java
char operation = '+';
int sum = 0;
switch(operation) {
    case '-':
        System.out.println("Subtraction not allowed");
        break;
    case '+':
        sum++;
        System.out.println("Current sum: " + sum);
        break;
    default:
        System.out.println("Current sum: " + sum);
        break;
}
```

BranchOrder.java

```java
char operation = '+';
int sum = 0;
switch(operation) {
    case '+':
        sum++;
        System.out.println("Current sum: " + sum);
        break;
    case '-':
        System.out.println("Subtraction not allowed");
        break;
    default:
        System.out.println("Current sum: " + sum);
        break;
}
```

BranchOrder.java

```
char operation = '+';
int sum = 0;
switch(operation) {
    case '+':
        sum++;
        System.out.println("Current sum: " + sum);
        break;
    default:
        System.out.println("Current sum: " + sum);
        break;
    case '-':
        System.out.println("Subtraction not allowed");
        break;
}
```

BranchOrder.java

```
char operation = '+';
int sum = 0;
switch(operation) {
    case '+':
        sum++;



    default:
        System.out.println("Current sum: " + sum);
        break;
    case '-':
        System.out.println("Subtraction not allowed");
        break;
}
```

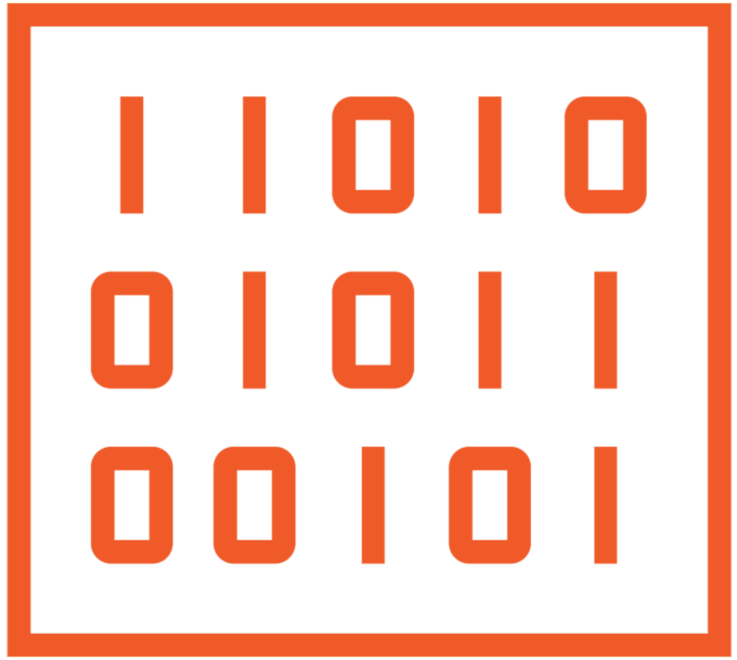
BranchOrder.java

```
char operation = ' ';
int sum = 0;
switch(operation) {
    case '+':
        sum++;



    default:
        System.out.println("Current sum: " + sum);
        break;
    case '-':
        System.out.println("Subtraction not allowed");
        break;
}
```

BranchOrder.java

```java
char operation = '-';
int sum = 0;
switch(operation) {
    case '+':
        sum++;



    default:
        System.out.println("Current sum: " + sum);
        break;
    case '-':
        System.out.println("Subtraction not allowed");
        break;
}
```

Switch Supported Types



All integral types except long

- char, byte, short, int

Supported integral type wrappers

- Character, Byte, Short, Integer

String

enum Values

Switch Supported Types

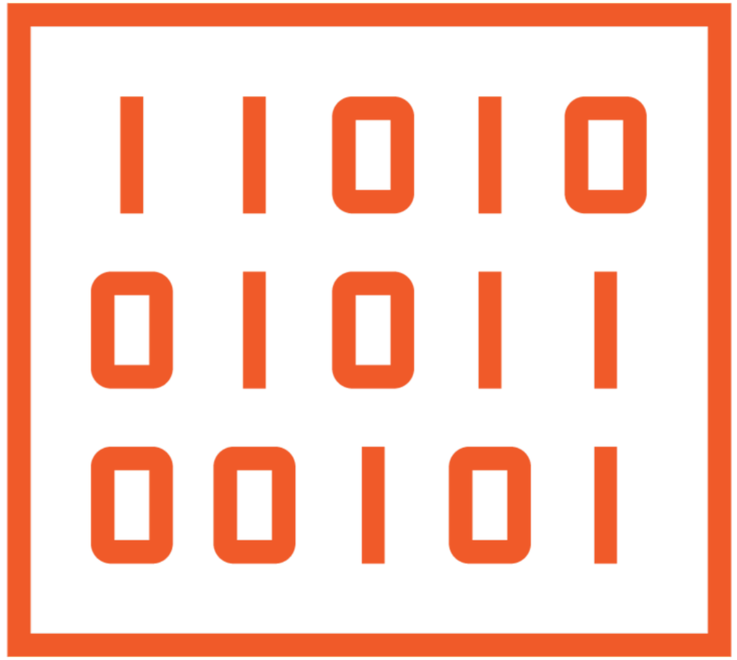
Day.java

```
enum Day {
    SUN, MON, TUE, WED, THU, FRI, SAT
}
```

Main.java

```
Day today = Day.SUN;
switch(today) {
    case SAT:
    case SUN:
        System.out.println("Weekend");
        break;
    default:
        System.out.println("Weekday");
        break;
}
```

Switch Supported Values



Switch test value

- Any expression that returns a value

Branch case value

- Any constant expression
- Value must be resolvable at compile time

Switch Supported Values

```
int iVal = 10;
      int evenValue = 0;

switch(iVal % 2) {
    case 0:
        System.out.println("even");
        break;
    case 1:
        System.out.println("odd");
        break;
}
```

Switch Supported Values

```
int iVal = 10;
      int evenValue = 0;
final int oddValue = 1;
switch(iVal % 2) {
    case evenValue:
        System.out.println("even");
        break;
    case 1:
        System.out.println("odd");
        break;
}
```

Switch Supported Values

```
int iVal = 10;
final int evenValue = 0;
final int oddValue = 1;
switch(iVal % 2) {
    case evenValue:
        System.out.println("even");
        break;
    case oddValue:
        System.out.println("odd");
        break;
}
```

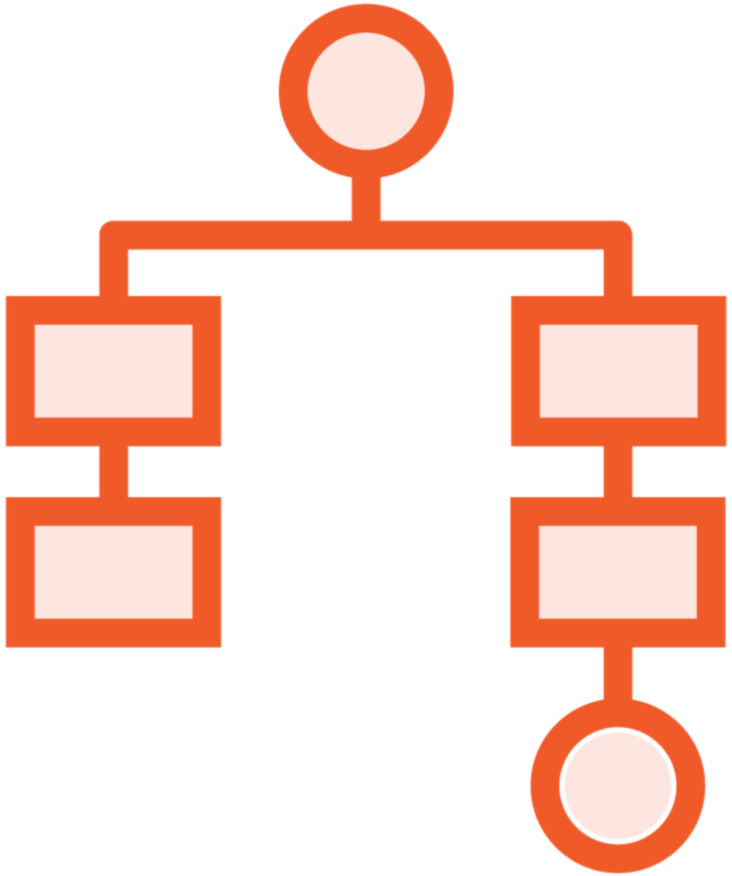
Switch Supported Values

```
int iVal = 10;
final int evenValue = 0;

switch(iVal % 2) {
    case evenValue:
        System.out.println("even");
        break;
    case evenValue + 1:
        System.out.println("odd");
        break;
}
```

Switch Supported Values

```
int iVal = 10;
final byte evenValue = 0;

switch(iVal % 2) {
    case evenValue:
        System.out.println("even");
        break;
    case evenValue + 1:
        System.out.println("odd");
        break;
}
```

Choosing Between Switch and if-else



if-else

- Extremely flexible
- Can handle most-any condition

Switch

- Test must be based on exact value match
- Type and value requirements
- Often allows intent to be more clearly expressed than if-else

Comparing Switch and if-else

```
if(operation == '+') {
    result = val1 + val2;
    System.out.println("add");
}
else if(operation == '-') {
    result = val1 - val2;
    System.out.println("subtract");
}
else {
    System.out.println("invalid operation");
}
```

Comparing Switch and if-else

```
switch(operation){
    case '+':
        result = val1 + val2;
        System.out.println("add");
        break;
    case '-':
        result = val1 - val2;
        System.out.println("subtract");
        break;
    default:
        System.out.println("invalid operation");
}
```

Comparing Switch and if-else

```
if(grade == 'A')
    System.out.println("Great work!");
else if(grade == 'B' || grade == 'C' || grade == 'D')
    System.out.println("You passed!");
else
    System.out.println("Try again next time");
```

Comparing Switch and if-else

```
switch(grade) {
    case 'A':
        System.out.println("Great work!");
        break;
    case 'B' or 'C' or 'D':
}
```

Comparing Switch and if-else

```
switch(grade) {
    case 'A':
        System.out.println("Great work!");
        break;
    case 'B':
    case 'C':
    case 'D':
        System.out.println("You passed!");
        break;
    default:
        System.out.println("Try again next time");
}
```

No Practical Switch Representation

Not testing for exact match

Not testing for exact match

```
if (age > 64) {
    System.out.println("Senior adult");
}
else if (age > 17) {
    System.out.println("Adult");
}
else {
    System.out.println("Minor");
}
```

No Practical Switch Representation

```
int items = 100;
final int maxItems = readMaxItems();
if (items == maxItems)
    System.out.println("Max reached");
else
    System.out.println("Still room");
```

Method call makes maxItems value unknowable at compile time

Summary

Switch statement

- Test value against multiple matches
- Transfer control based on match

When match found

- Starts running code in that branch
- Continues executing following code until switch exited

Break

- Immediately exists switch
- Often used at the end of each branch

Summary

Branch order

- Does not affect branch selection
- Order may be important if execution allowed to flow across branches

Supported data types

- char, byte, short, int
- Character, Byte, Short, Integer
- String
- enum values

Summary

Switch test value

- Any expression that returns a value

Branch case value

- Any constant expression
- Must be resolvable at compile time