02

Utilize o switch

Suponha que um programa tenha que reagir diferentemente para três casos possíveis. Por exemplo, suponha que o usuário possa passar três valores possíveis: `1` , `2` e `3` . Se for `1` , o programa deve imprimir "PRIMEIRA OPCAO", se for `2` , "SEGUNDA OPCAO", e se for `3` , "TERCEIRA OPCAO".

Isso pode ser implementado com `if/else` . Mas, há uma outra possibilidade. O Java, assim como outras linguagens de programação, oferece o comando **switch**. Ele permite testar vários casos de uma maneira diferente do `if/else` .

```
int opcao = 1;
switch (opcao) {
    case 1:
        System.out.println("PRIMEIRA OPCAO");
    case 2:
        System.out.println("SEGUNDA OPCAO");
    case 3:
        System.out.println("TERCEIRA OPCAO");
}
```

COPIAR CÓDIGO

O `switch` tem uma sintaxe cheia de detalhes e uma semântica pouco intuitiva. Vamos analisar cada um desses detalhes separadamente para ficar mais simples.

O argumento do `switch` dever ser uma variável compatível com o tipo primitivo `int` , um ::wrapper:: de um tipo menor que `Integer` , uma **String** ou um **enum**.

Enums Não são cobrados nessa prova, então vamos focar apenas nos outros dois casos.

O valor de cada `case` deve ser compatível com o tipo do argumento do `switch`, caso contrário será gerado um erro de compilação na linha do `case` inválido.

```
//argumento do switch int, e cases int
int valor = 20;
switch (valor){
    case 10 : System.out.println(10);
    case 20 : System.out.println(20);
}

//Argumento String, e cases String
String s = "Oi";
switch (s) {
    case "Oi": System.out.println("Olá");
    case "Hi": System.out.println("Hello");
}

//Argumento Byte, e cases byte
Byte b = 10;
switch (b) {
    case 10: System.out.println("DEZ");
}

//argumento do switch int, e cases string, não compila
int mix = 20;
switch (mix){
    case "10" : System.out.println(10);
    case "20" : System.out.println(20);
}
```

COPIAR CÓDIGO

Cuidado pois `switch` de `double` não faz sentido conforme a lista de argumentos que citamos compatíveis com o `switch` !

```
double mix = 20;
switch (mix){  // não compila
    case 10.0 : System.out.println(10);
    case 20.0 : System.out.println(20);
}
```

COPIAR CÓDIGO

Você pode usar qualquer tipo primitivo menor que um `int` como argumento do `switch` , desde que os tipos dos cases sejam compatíveis:

```
//argumento do switch byte
byte valor = 20;

switch (valor){
    // Apesar de ser inteiro, 10 cabe em um byte, o compilador
    // fará o cast automaticamente
    case 10 :
        System.out.println(10);
}

switch (valor){
    // Neste caso, o número é muito grande, o compilador não
    // fará o cast e teremos um erro de compilação pois os
tipos
    // são incompatíveis
    case 32768 :  //erro
        System.out.println(10);
}
```

COPIAR CÓDIGO

Em cada `case`, só podemos usar como valor um literal, uma variável `final` atribuída com valor literal, ou expressões envolvendo os dois. Nem mesmo `null` é permitido:

```
int valor = 20;
final int CINCO = 5;
int trinta = 30;

switch (valor) {
    case CINCO: // constante
        System.out.println(5);
    case 10: // literal
        System.out.println(10);
    case CINCO * 4: // operação com constante e literal
        System.out.println(20);
    case trinta: // erro, variável
        System.out.println(30);
    case trinta + CINCO: //erro, operação envolvendo variável
        System.out.println(35);
    case null: // erro, null em case
        System.out.println("null");
}
```

COPIAR CÓDIGO

Constantes em cases

Para ser considerada uma constante em um `case`, a variável, além de ser final, também deve ter sido inicializada durante a sua declaração. Inicializar a variável em outra linha faz com que ela não possa ser usada como valor em um `case`:

```
int v = 10;
final int DEZ = 10;
final int VINTE; // final, mas não inicializada
VINTE = 20; // inicializada

switch (v) {
case DEZ:
    System.out.println("DEZ!");
    break;
case VINTE: //erro
    System.out.println("DEZ!");
    break;
}
```

COPIAR CÓDIGO

O `switch` também aceita a definição de um caso padrão, usando a palavra `default`. O caso padrão é aquele que deve ser executado se nenhum `case` "bater".

```
int opcao = 4;
switch (opcao) {
    case 1:
        System.out.println("PRIMEIRA OPCAO");
    case 2:
        System.out.println("SEGUNDA OPCAO");
    case 3:
        System.out.println("TERCEIRA OPCAO");
    default:
        System.out.println("CASO PADRAO");
}
```

COPIAR CÓDIGO

Um detalhe sobre a sintaxe do `default` é que ele pode aparecer antes de um ou de diversos `case` s. Desta forma:

```
int opcao = 4;
switch(opcao) {
    case 1:
        System.out.println("PRIMEIRA OPCAO");
    case 2:
        System.out.println("SEGUNDA OPCAO");
    default:
        System.out.println("CASO PADRAO");
    case 3:
        System.out.println("TERCEIRA OPCAO");
}
```

COPIAR CÓDIGO

Um comportamento contraintuitivo do `switch` é que, quando executado, se algum `case` "bater", tudo que vem abaixo é executado também, todos os `case` s e o `default` , se ele estiver abaixo. Esse comportamento também vale se cair no `default` . Por exemplo, o código anterior imprime:

```
CASO PADRAO
TERCEIRA OPCAO
```

COPIAR CÓDIGO

Com esse comportamento, podemos inclusive criar `cases` sem nenhum bloco de código dentro:

```
int v = 1;
switch(v){
    case 1:
    case 2:
    case 3:
        System.out.println("Até 3");
}
```

COPIAR CÓDIGO

Para mudar esse comportamento e não executar o que vem abaixo de um `case` que bater ou do `default`, é necessário usar o comando `break` em cada `case`.

```
int opcao = 4;
switch(opcao) {
    case 1:
        System.out.println("PRIMEIRA OPCAO");
        break;
    case 2:
        System.out.println("SEGUNDA OPCAO");
        break;
    default:
        System.out.println("CASO PADRAO");
        break;
    case 3:
        System.out.println("TERCEIRA OPCAO");
        break;
}
```

COPIAR CÓDIGO

Neste caso, só será impresso "CASO PADRAO".