

2-4.制御文(条件分岐)

[提出する](#)[評価を受ける](#)

制御文（条件分岐）[if, else, else if, switch]

Step1: 概念を知る

制御構文は一言で言うと、**プログラムを制御する文のこと**です。

そして、制御構文は「分岐」と「繰返し」の2つに分類できます。

「**分岐**」には、以下の2つの構文、

- if文（+ else, else if文）
- switch文
-

「**繰返し**」には以下の3つの構文が存在します。

- for文
- while文
- do-while文

以下より、**if文**、**switch文** に分けて解説していきます。

Step2-1: 【分岐の制御①】 if文

if文 は、括弧の中の条件によって処理の流れを分岐させます。

以下の文では、括弧の中の条件式の判定結果が **true**（真） の場合のみ、**{ }** の中に記述されている処理が実行されます。

条件式の判定結果が **false**（偽） の場合、**{ }** の中の処理は実行されません。

```
if (条件式) {  
    /* ... 条件式が true の場合に行う処理 ... */  
}
```

Step2-2: else文

else文 は、「それ以外」の場合に実行する処理を記述します。

つまり、if文の条件式の判定結果が **false** だった場合に、else文の **{ }** 内の処理が実行されます。

```
if (条件式) {  
    /* ... 条件式が true の場合に行う処理 ... */  
} else {  
    /* ... 条件式が true でなかった場合 (=falseの場合) に行う処理 ... */  
}
```

?

Step2-3: else if文

else-if文 は、「それ以外」で更に別の条件を判定する場合に使用します。

つまり、if文の条件式の判定結果が `false` だった場合に、else-if文の条件式の判定が行われます。

そして、else-if文の条件式の判定結果が `true` だった場合に、else-if文の `{ }` 内の処理が実行されます。

```
if (条件式1) {  
    /* ... 条件式1が true の場合に行う処理 ... */  
} else if (条件式2) {  
    /* ... 条件式1が false で、条件式2が true の場合に行う処理 ... */  
}
```

Step2-4: if, else, else if文（組み合わせ）

if文、else-if文、else文を組み合わせると、以下のようになります。

```
if (条件式1) {  
    /* ... 条件式1が true の場合に行う処理 ... */  
} else if (条件式2) {  
    /* ... 条件式1が false で、条件式2が true の場合に行う処理 ... */  
} else if (条件式3) {  
    /* ... 条件式1と条件式2が false で、条件式3が true の場合に行う処理 ... */  
} else {  
    /* ... 上記の条件式1～3が全て false の場合に行う処理 ... */  
}
```

例題

【if文を使ったプログラム例】

下のプログラムでは、飲酒可能かどうかの判定をしています。

もし20歳以上だった場合は「酒が飲めるぞ！」と表示します。

もし19歳だった場合は「あと1年で酒が飲める！！」と表示します。

それ以外（18歳以下だった場合）は「まだまだ酒は飲めない・・・」と表示するプログラムです。

今回は **old** 変数に **25**（25歳）を代入しています。

```
public class Main {  
    public static void main(String[] args) {  
        // int型の変数oldに25を代入  
        int old = 25;  
  
        // もし、20歳以上の場合は、"酒が飲めるぞ！"と表示  
        if (old >= 20) {  
            System.out.println("酒が飲めるぞ！");  
        }  
  
        // でももし、19歳の場合は、"あと1年で酒が飲める！！"と表示  
        } else if (old == 19) {  
            System.out.println("あと1年で酒が飲める！！");  
        }  
  
        // それ以外の場合は、"まだまだ酒が飲めない・・・"と表示  
        } else {  
            System.out.println("まだまだ酒は飲めない・・・");  
        }  
    }  
}
```

解説

今回のif文の条件(`old >= 20`)は、

```
old > 20 && old == 20
```

という条件を省略した書き方です。

※&&については **Step2-5: 複数条件の分岐の論理演算子** の項目で説明しております。

【実行結果】

酒が飲めるぞ！

【if文を使ったプログラム例(文字列の比較)】

下のプログラムでは、文字列の比較をしています。

もし国名がAmericaだった場合は「アメリカ合衆国」と表示します。

もし国名がJapanだった場合は「日本」と表示します。

それ以外は「その他」と表示するプログラムです。

今回はStringクラスの変数にAmericaを代入しています。

※Stringクラスの比較の場合、==ではなくequalsを使用する必要があります。

```
public class Main {  
    public static void main(String[] args) {  
        // Stringクラスの変数countryに"America"を代入  
        String country = "America";  
  
        // もしcountryが"America"の場合、"アメリカ合衆国"と表示  
        if (country.equals("America")) {  
            System.out.println("アメリカ合衆国");  
        }  
  
        // でももし、countryが"Japan"の場合、"日本"と表示  
        } else if (country.equals("Japan")) {  
            System.out.println("日本");  
        }  
  
        // それ以外の場合、"その他"と表示  
        } else {  
            System.out.println("その他");  
        }  
    }  
}
```

【実行結果】

アメリカ合衆国

文字列の比較の場合、==ではなくequalsメソッドを使わなければならない理由は少々複雑で、そういうものだと覚えていてもひとまず問題はありますが、詳しく説明されている参考サイトを紹介しますので良ければ読んでみて下さい。

参考サイト

Step2-5: 複数条件の分岐

条件文を複数にしたいときもあります。

例えば、年齢が25歳未満で学生であれば、学割バックが使えます、としましょう。

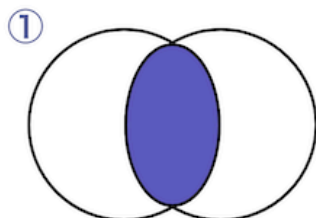
```
// 年齢  
int age = 24;  
  
// 学生かどうか  
boolean is student = true;
```

複数条件を指定する場合には論理演算子を使用します。

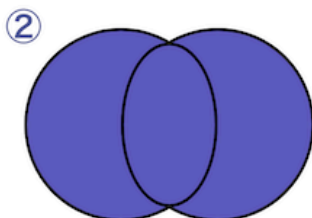
演算	演算記法	画像番号	説明
AND 演算	A && B	①	AとB両方の条件を満たす場合に true
OR 演算	A B	②	AまたはBどちらかの条件を満たす場合に true
NOT 演算	!A	③	Aではない場合に true

◆◆ 論理演算子 (Logical Operator) ◆◆

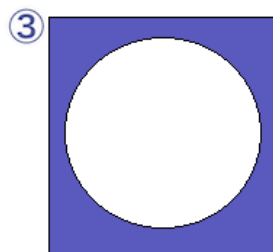
AND (A && B)



OR (A || B)



NOT (!A)



&&, ||が代表的です。

|の記号は馴染みがないかもしれませんが、Shift + ¥で出せます。(設定を変更していなければ)。

それでは、年齢が25歳未満で学生であれば、学割パックが使えます、という条件分岐を書いてみます。

```
public class Main {  
    public static void main(String[] args) {  
        // 年齢  
        int age = 24;  
  
        // 学生かどうか  
        boolean is_student = true;  
  
        // もし、年齢が25歳未満 かつ 学生 の場合、"学割パックが使えるよ"と表示  
        if (age < 25 && is_student) {  
            System.out.println("学割パックが使えるよ");  
        }  
    }  
}
```

【実行結果】

学割パックが使えるよ

解説

&&を使用することで、両方の条件を満たしていなければ「正しい」と判断しません。

逆に||を使用することで、どちらか片方だけでも満たしていれば「正しい」と判断します。

実際に試して確認してみてください。

Step2-6: else if

さて、年齢が25歳未満で学生であれば、学割パックが使えますが、
学生でなくても25歳未満であれば、若者応援割引が使えるとします。

この場合、下記のように続けてかくことができます。

```
public class Main {  
    public static void main(String[] args) {  
        // 年齢  
        int age = 24;  
  
        // 学生かどうか  
        boolean is_student = false;  
  
        // もし、年齢が25歳未満 かつ 学生 の場合、"学割バックが使えるよ"と表示  
        if (age < 25 && is_student) {  
            System.out.println("学割バックが使えるよ");  
        }  
  
        // でももし、年齢が25歳未満の場合、"若者応援割引が使えるよ"と表示  
        } else if (age < 25) {  
            System.out.println("若者応援割引が使えるよ");  
        }  
    }  
}
```

【実行結果】

若者応援割引が使えるよ

解説

このelse ifはいくつでも書くことができます。

もし全ての条件に当てはまらなかったときの処理を記述する場合はelseを使用します

Step3: 【分岐の制御②】 switch文

switch文は、括弧内の式によって処理の流れを分岐します。

ひとつの式の結果から多方向に分岐する、多分岐処理で使用します。

主な特徴、記述方法は以下のとおりです。

1. 複数のcaseという選択肢を記述し分岐させる
2. 各々のcaseの最後にbreak文を記述することで、処理中の caseを終了してswitch文から抜ける
3. どのcase選択肢にも当てはまらない場合は、defaultを記述することで、if文の else と同等の機能を持たせる

以下に実際の書き方を提示します。

```
switch (式) {  
    case 値1:   
        /* ... 式が 値1 の場合に行う処理 ... */  
        break;  
  
    case 値2:   
        /* ... 式が 値2 の場合に行う処理 ... */  
        break;  
  
    default:   
        /* ... 式が上記case選択肢のどれにも当てはまらない場合に行う処理 ... */  
        break;  
}
```

switch文の分岐には数値や文字が用いられることが多いですが、ここでは数値で分岐させているプログラム例を以下に示します。

例題

【switch文を使ったプログラム例】

```
public class Main {  
    public static void main(String[] args) {  
        int kubun = 2;  
  
        switch (kubun) {  
            case 1:  
                System.out.println("kubunが1のときの処理を行います。");  
                break;  
  
            case 2:  
                System.out.println("kubunが2のときの処理を行います。");  
                break;  
  
            case 3:  
                System.out.println("kubunが3のときの処理を行います。");  
                break;  
  
            case 9:  
                System.out.println("kubunが9のときの処理を行います。");  
                break;  
  
            default:  
                System.out.println("kubunが想定外の値のときの処理を行います。");  
                break;  
        }  
    }  
}
```

【実行結果】

kubunが2のときの処理を行います。

(JDK1.7以降)
式及び値に文字列型が使用できるようになりました。

```
public class Main {  
    public static void main(String[] args) {  
        String color = "YELLOW";  
  
        switch (color) {  
            case "RED":  
                System.out.println("colorがREDのときの処理を行います。");  
                break;  
  
            case "YELLOW":  
                System.out.println("colorがYELLOWのときの処理を行います。");  
                break;  
  
            case "GREEN":  
                System.out.println("colorがGREENのときの処理を行います。");  
                break;  
  
            default:  
                System.out.println("colorが想定外の値のときの処理を行います。");  
                break;  
        }  
    }  
}
```

【実行結果】

colorがYELLOWのときの処理を行います。

提出課題

提出ファイル

・ Task2_4.java
2-4/Task2_4.java に課題がありますので、指示に従って記述して下さい。

評価概要

学生から秘匿	No
参加者	80

提出	53
要評定	0