



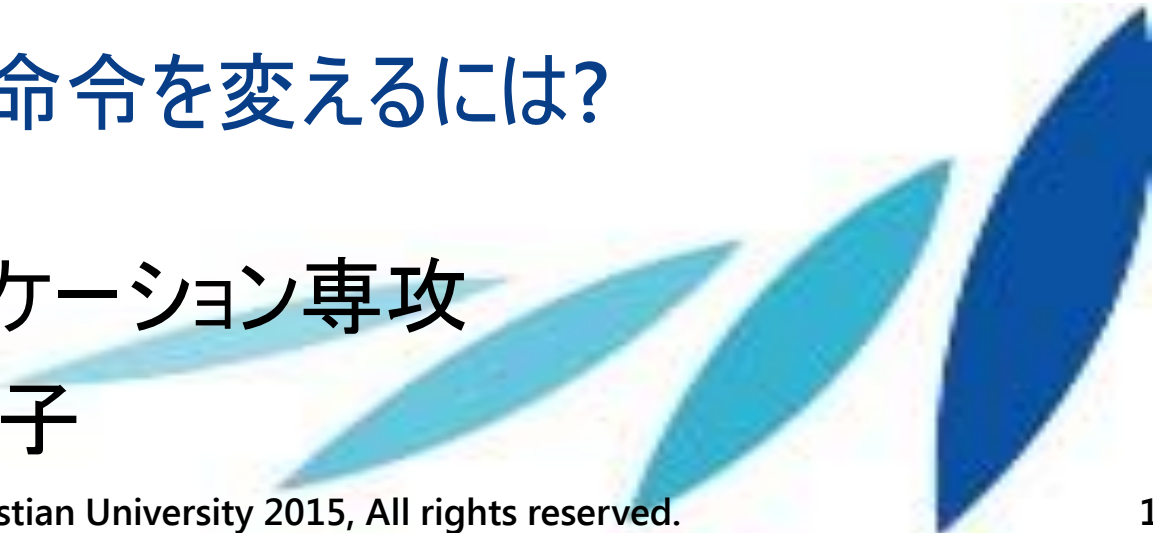
情報処理技法 (Javaプログラミング)1

第6回

条件によって実行させる命令を変えるには？

人間科学科コミュニケーション専攻

白銀 純子





第6回の内容

⌘ 演算子

⌘ boolean型

⌘ 条件分岐





前回の復習問題の解答

⌘ String型の変数「message」に、「おつりは"1234"円です。」という文字列を代入したいとし、「1234」という金額が変数「change」に入っている場合、変数「message」への代入の文を作成しなさい。

⌘ message = 代入文

※「代入文」の部分を作成すること

おつりは"1234"円です。 ➡ おつりは" 1234 "円です。

➡ おつりは¥" change ¥"円です。

➡ "おつりは¥"" + change + "¥"円です。"

解答例: "おつりは¥"" + change + "¥"円です。"



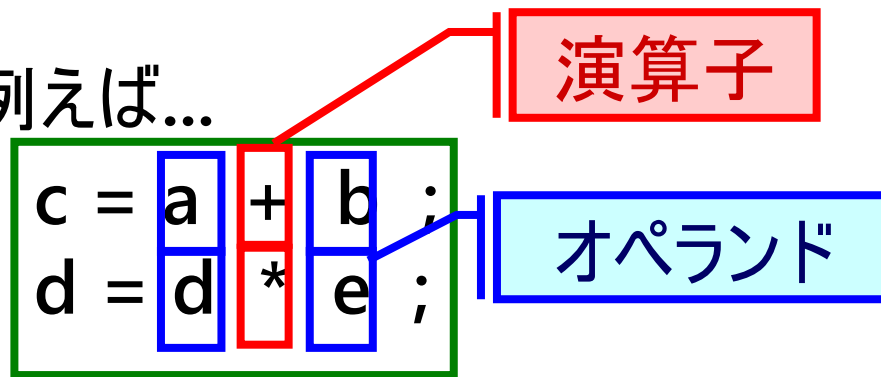
演算子



式とは

⚙️ 演算子とオペランドを組み合わせて何らかの結果を求めるもの

例えば...



- 演算子: その式で何をするかを定めるもの
- オペランド: その式で処理される変数や値



演算子(1)(p. 66)

演算子	使い方	意味
+	$a + b$	$a + b$ の結果を求める
-	$a - b$	$a - b$ の結果を求める
*	$a * b$	$a * b$ の結果を求める
/	a / b	a / b の商を求める
%	$a \% b$	a / b の余りを求める

※ a と b は変数または決まった値





演算子(2)(p. 146)

演算子	使い方	意味
>	$a > b$	$a > b$ が正しいかどうかを求める
>=	$a \geq b$	$a \geq b$ が正しいかどうかを求める
<	$a < b$	$a < b$ が正しいかどうかを求める
<=	$a \leq b$	$a \leq b$ が正しいかどうかを求める
==	$a == b$	a と b が等しいかどうかを求める
!=	$a != b$	a と b が等しくないかどうかを求める

※ a と b は変数または決まった値



演算子(3)(p. 146)

演算子	使い方	意味
&&	式1 && 式2	式1と式2がどちらも正しいかどうかを求める
	式1 式2	式1と式2のどちらかが正しいかどうかを求める

例えば...

`a > b && c < d`

a > bが正しく、かつ c < dも正しいかどうかを求めている

➡ a > bが正しくかつ c < d も正しい場合は、結果は「yes」
どちらか一方が正しくない場合は、結果は「no」



boolean型(p. 146)

- ⊗ 「正しい」か「正しくない」かの結果を表すデータ型
 - ⊗ intやdoubleと同様のデータ型
 - ⊗ 値は2種類のみ
- ⊗ 「正しい」場合には「**true**」という値
- ⊗ 「正しくない」場合には「**false**」という値

例えば...

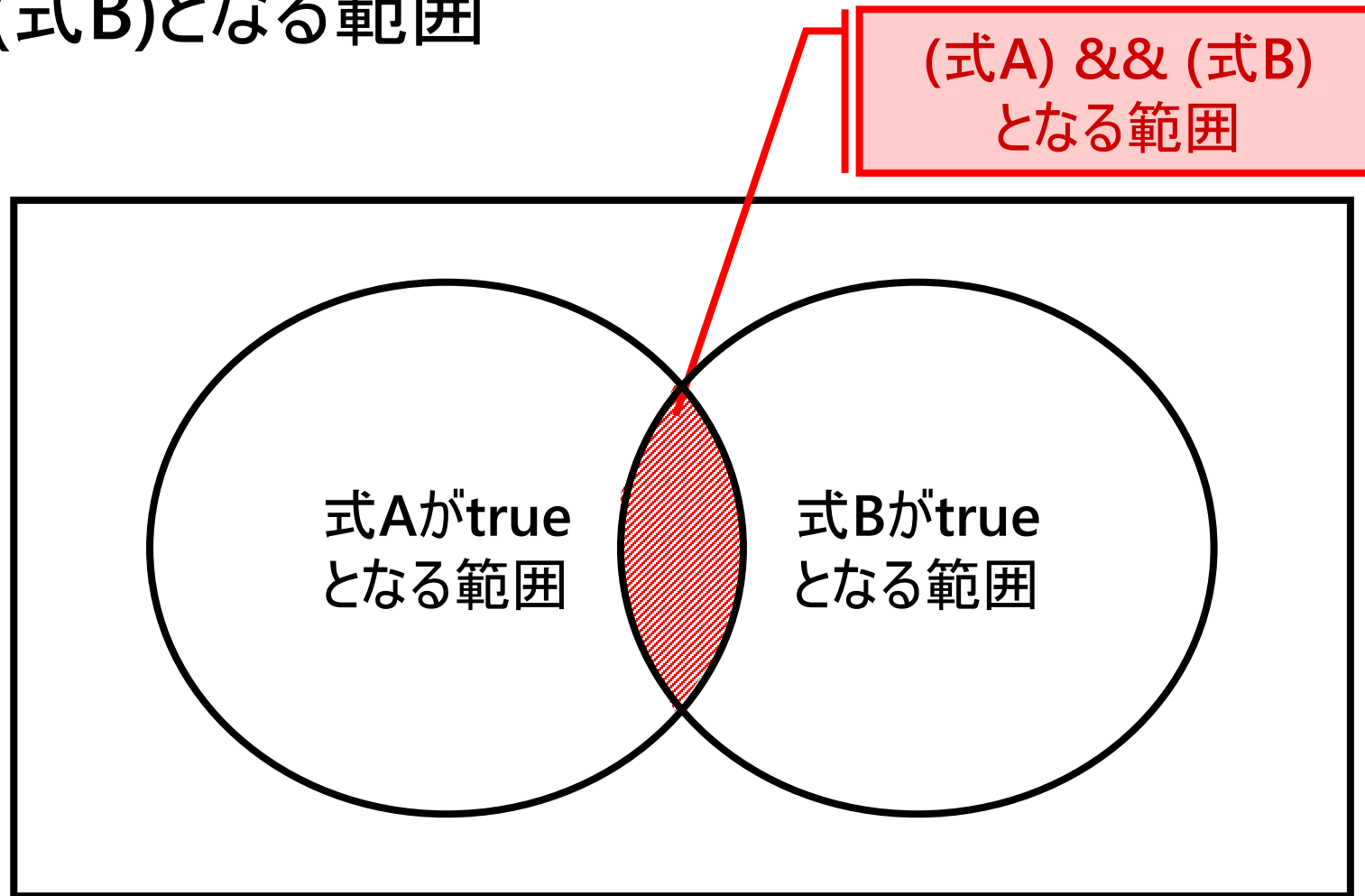
`a > b` (aはbより大きい)
`a != b` (aとbは違う値である)
`a == 10` (aの値は10である)
`a < b && c >= d`
(aはbより小さく、かつcはd以上である)

- 
- 正しいければ「**true**」
 - 正しくなければ「**false**」



boolean型で結果出す演算子(1)(p. 147)

⌘ (式A) && (式B)となる範囲



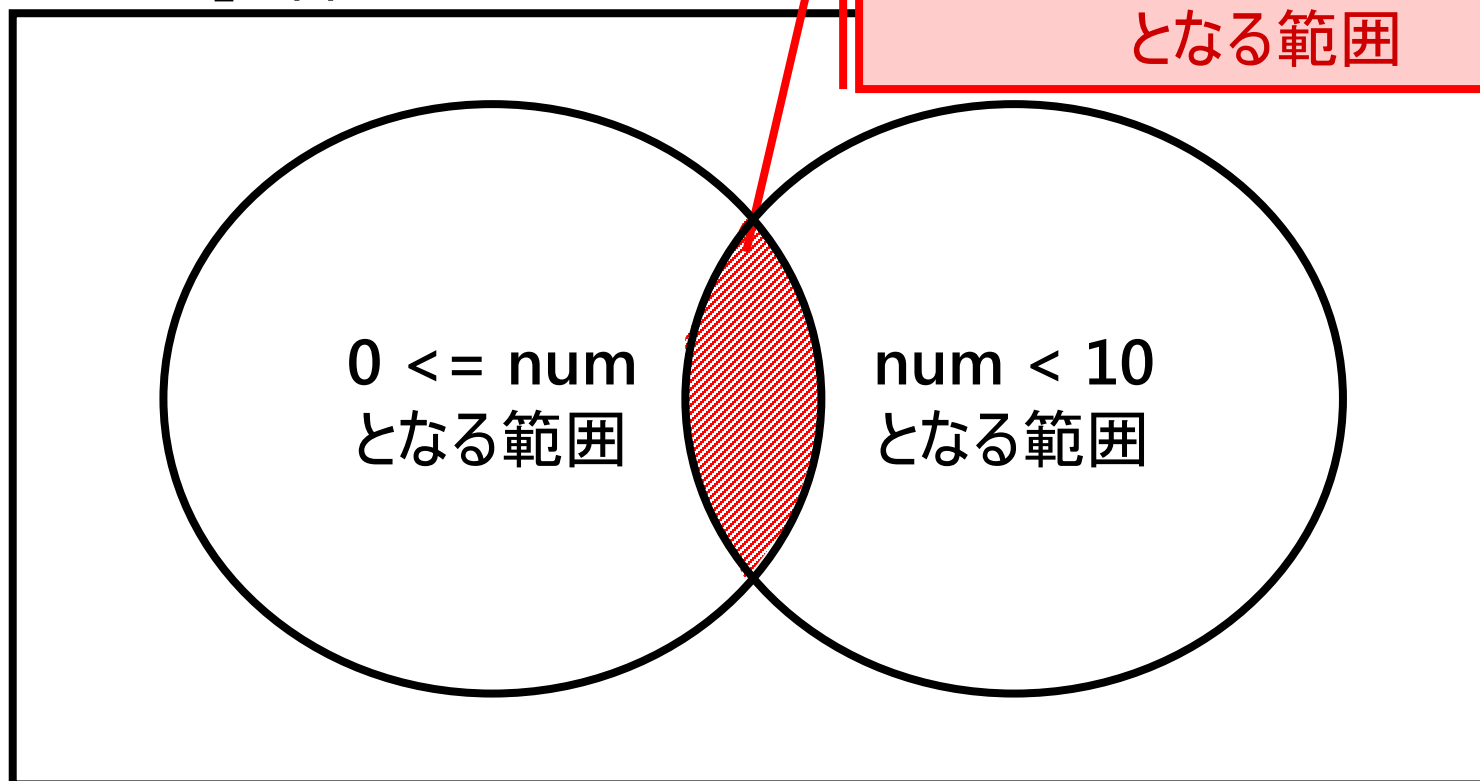
boolean型で結果出す演算子(2)(p. 147)

❁ Ex. $0 \leq \text{num} < 10$ の表現

❁ numはint型の変数

❁ $(0 \leq \text{num}) \ \&\& \ (\text{num} < 10)$ と書く

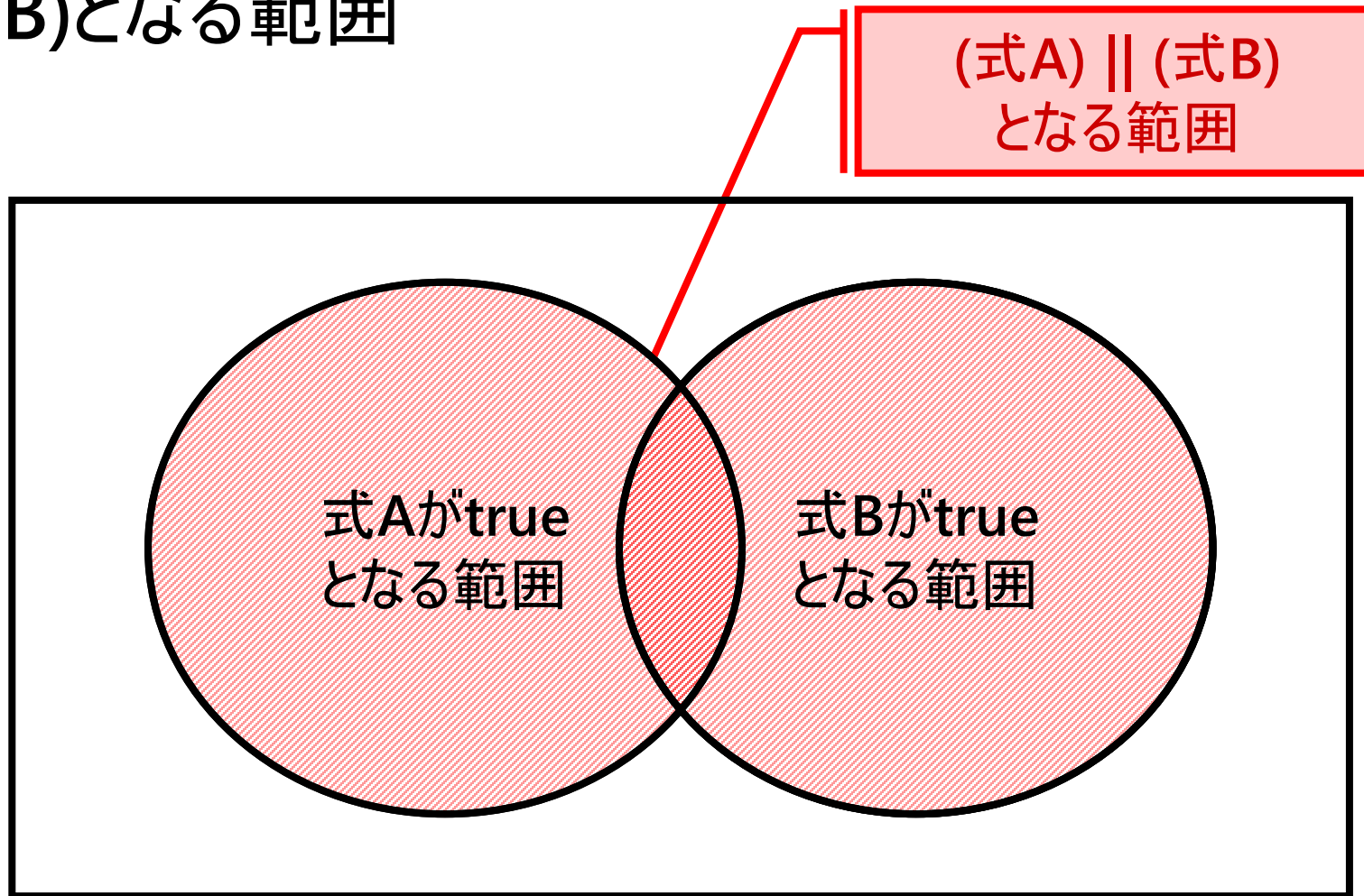
❁ 「 $0 \leq \text{num} < 10$ 」と書いてはいけない



$(0 \leq \text{num}) \ \&\& \ (\text{num} < 0)$
となる範囲

boolean型で結果出す演算子(3)(p. 147)

⊗ (式A) || (式B)となる範囲



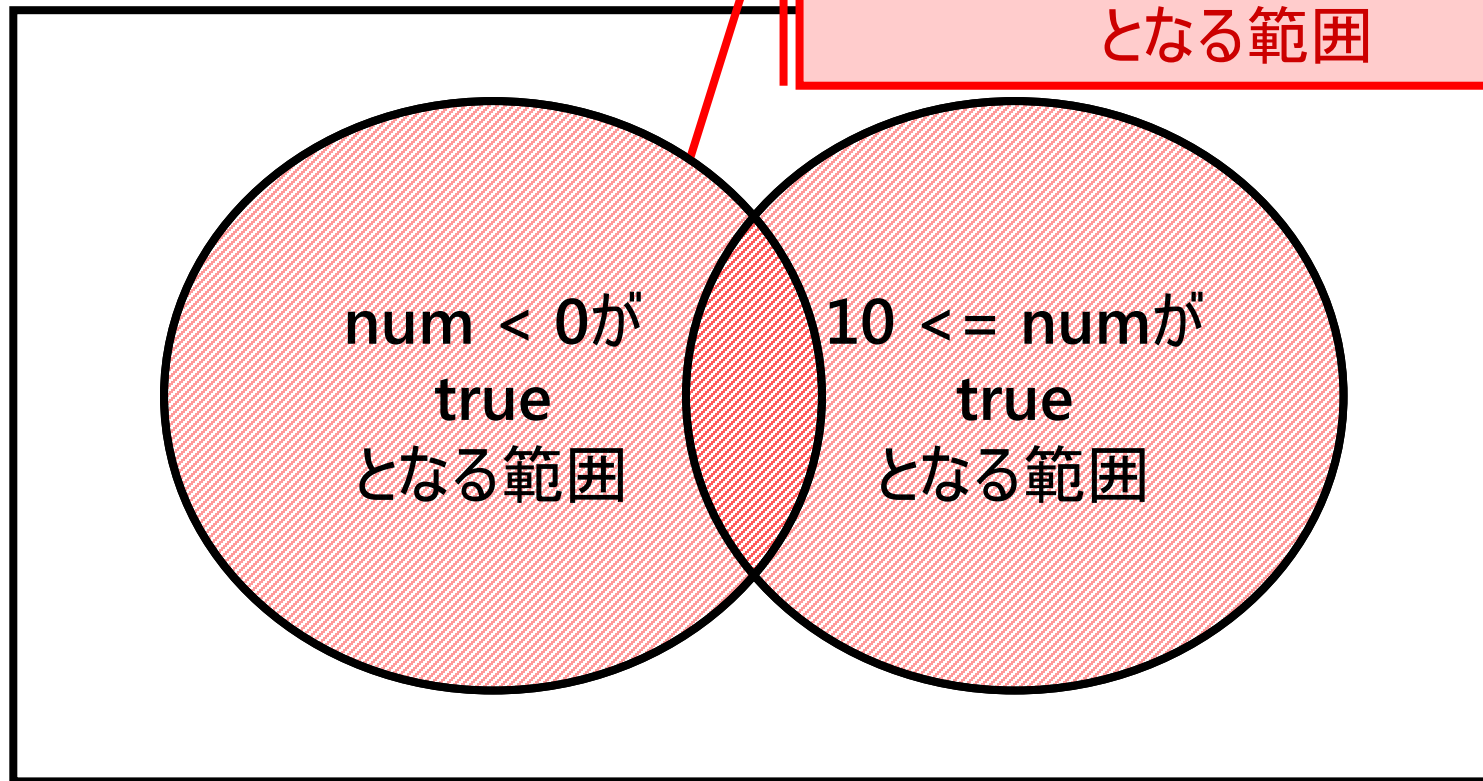
boolean型で結果出す演算子(4)(p. 147)

❁ Ex. $\text{num} < 0$ または $10 \leq \text{num}$ の表現

❁ num は int 型の変数

❁ $(\text{num} < 0) \parallel (10 \leq \text{num})$ と書く

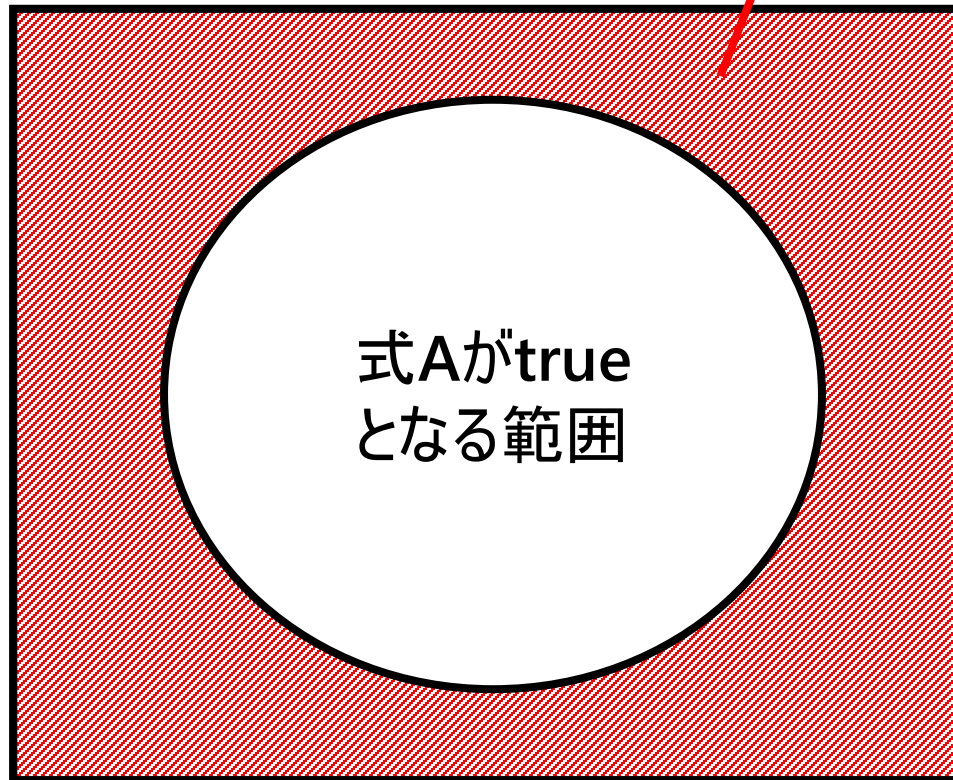
$(\text{num} < 0) \&\& (10 \leq \text{num})$
となる範囲



boolean型で結果出す演算子(5)(p. 147)

⊗!(式A)となる範囲

!(式A) となる範囲

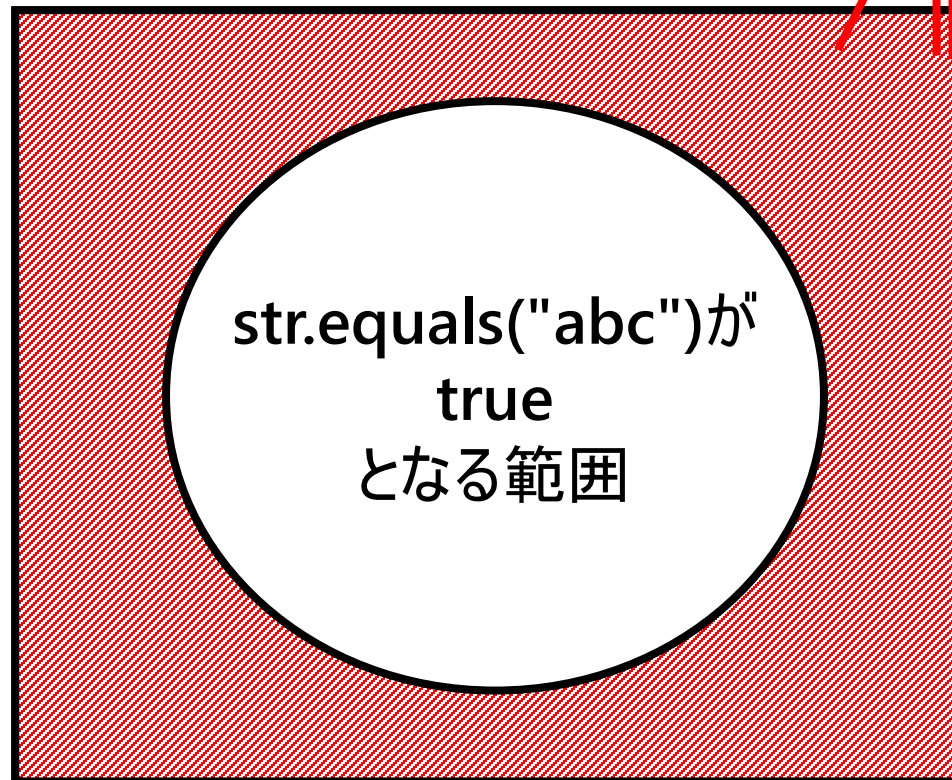


boolean型で結果出す演算子(6)(p. 147)

⊗ Ex. `str != "abc"` の表現

⊗ `str`はString型の変数

⊗ `!(str.equals("abc"))` と書く



`!(str.equals("abc"))`
となる範囲



条件分岐





状況によってすることが違う場合 (p. 152)

❁ いろいろな状況によって、したいことが違うこともある!

例えば... 予算5000円で服を買いたい!

買いたい服の値段が5000円以下

➡ 服を買う

買いたい服の値段が5000円より高い

➡ 服を買うのをあきらめる

もし...ならば～をする ➡ プログラムではどうする?





プログラムで、「もし...ならば～」(p. 152)

条件分岐と呼ぶ

⌘「if」を使う(「if文」と呼ぶ)

⌘「switch」を使う(「switch」文と呼ぶ)

⌘ switch文は、どのような形のものでもif文に書き換えることができる

⌘ if文は、switch文に書き換えることができないものが多い

条件分岐には、switch文よりもif文を使うことが多い





条件分岐(その1)(1)(p. 152)

⊛もし...ならば、～をする

boolean型で
結果が出るもの

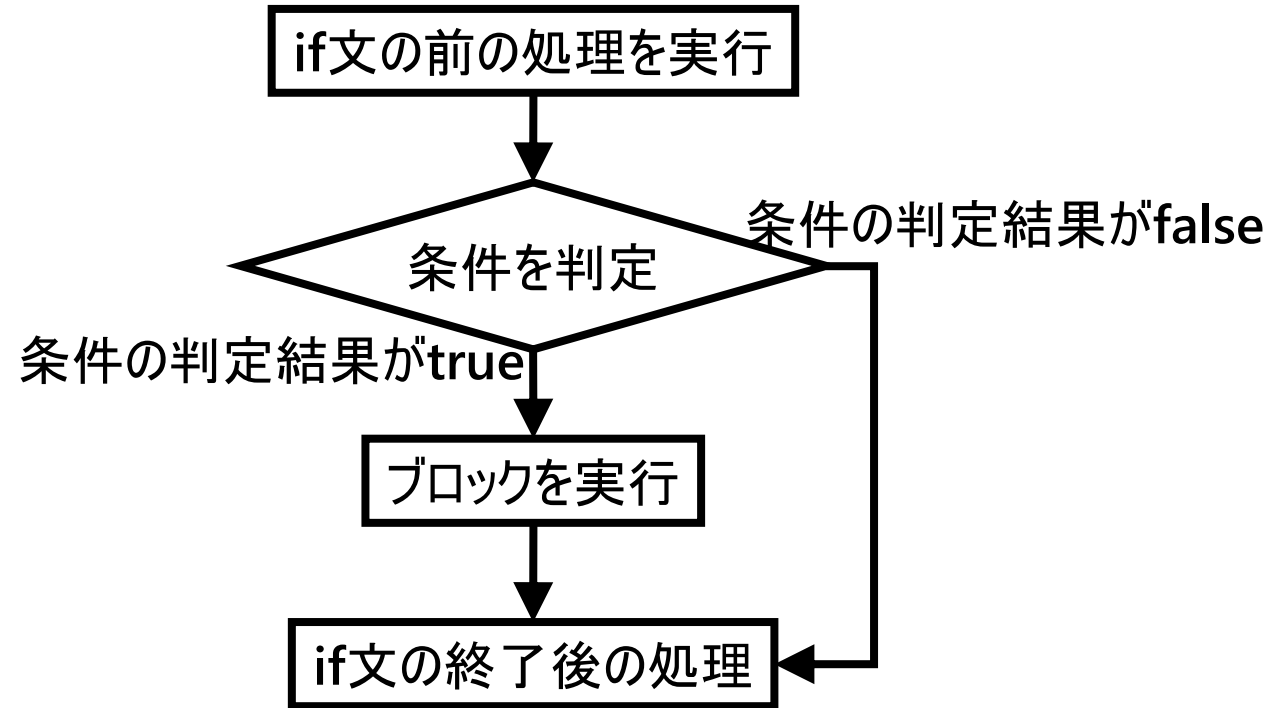
「条件」がtrueのときにすることは、
このカッコ内に書くこと

```
if (条件) {  
    「条件」がtrueのときにすること  
}
```

条件分岐(その1)(2)(p. 152)

```
if (条件) {  
  文1;  
  文2;  
  文3;  
}
```

ブロック





条件分岐(その1)(例)(p. 152)

例: 買いたい服の値段が5000円以内ならば、服を買う



```
if (買いたい服の値段 <= 5000円) {  
    服を買う;  
}
```

服の値段の変数を「cloth」(int型)とすると...

```
if (cloth <= 5000) {  
    服を買う;  
}
```





条件分岐(その2)(1)(p. 153)

⌘ もし...ならば～をし、そうでなければ---をする

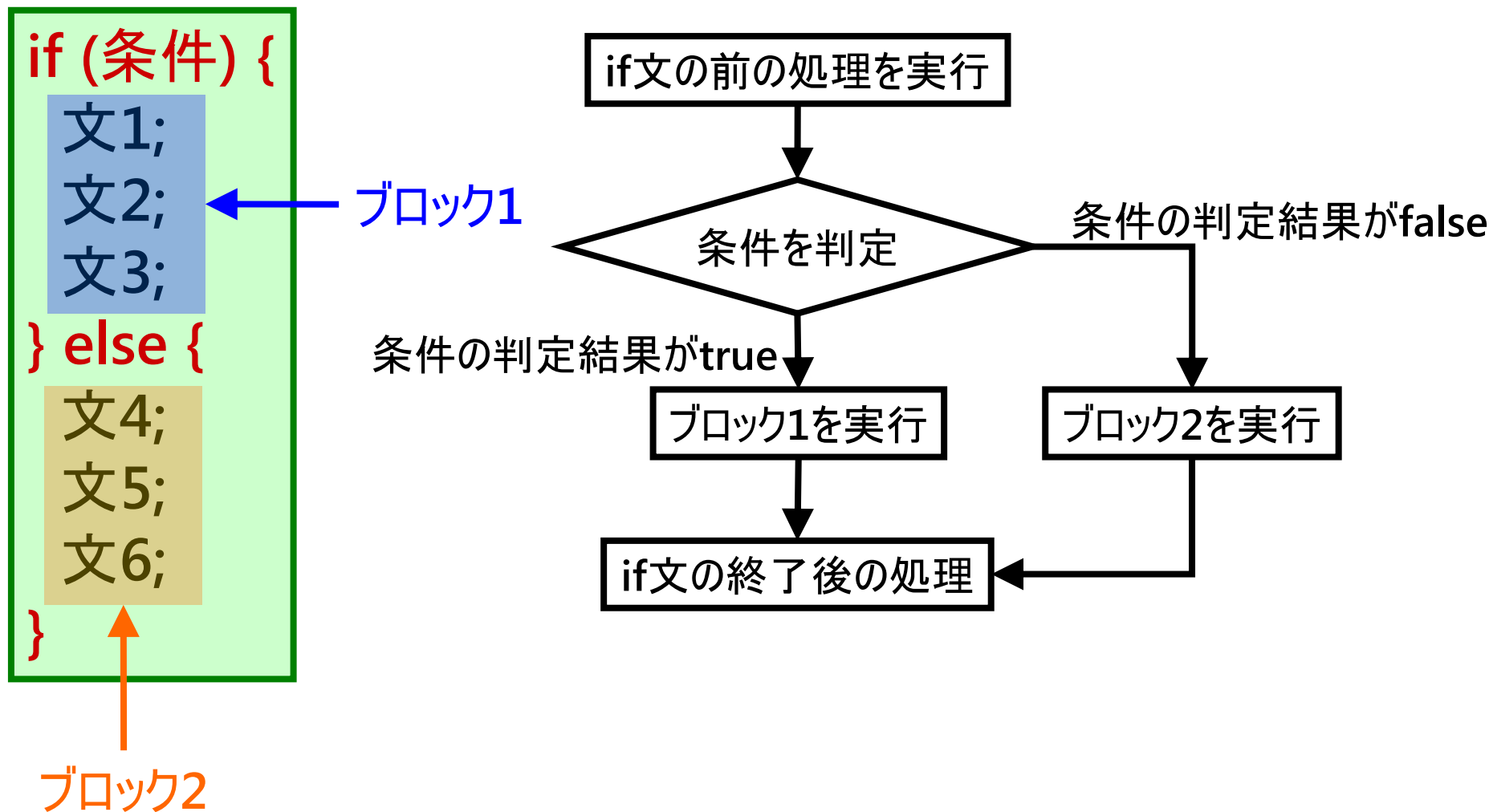
```
if (条件) {  
    「条件」がtrueのときにすること  
}  
else {  
    「条件」がfalseのときにすること  
}
```



「else」の部分が
実行される範囲



条件分岐(その2)(2)(p. 153)



条件分岐(その2)(例)(p. 153)

例: 買いたい服の値段が5000円以内ならば
服を買い、そうでなければ服を買うのをあきらめる



```
if (買いたい服の値段 <= 5000円) {  
    服を買う;  
} else {  
    服を買うのをあきらめる;  
}
```

買いたい服の値段 > 5000円
のときにすること

服の値段の変数を
「cloth」(int型)とすると..

```
if (cloth <= 5000) {  
    服を買う;  
} else {  
    服を買うのをあきらめる;  
}
```


条件分岐(発展)(1)(p. 156)

⚙️ もしAならばXをし、AでなくてBならばYをし、
AでもBでもなければZをする

```
if (条件A) {  
    Xをする;  
} else if (条件B) {  
    Yをする;  
} else {  
    Zをする;  
}
```

- まず最初に「条件A」を判断し、「条件A」が正しければXをする。
- 「条件A」が正しくなければ「条件B」を判断し、「条件B」が正しければYをする。

➡ AとBがどちらも正しい場合でも、Xをする(Yはしない)

※「else if」で、いくつでも条件をつけることができる

条件分岐(発展)(2)(p. 156)

```
if (条件A) {
```

```
  文1;  
  文2;  
  文3;
```

← ブロック1

```
} else if (条件B) {
```

```
  文4;  
  文5;  
  文6;
```

← ブロック2

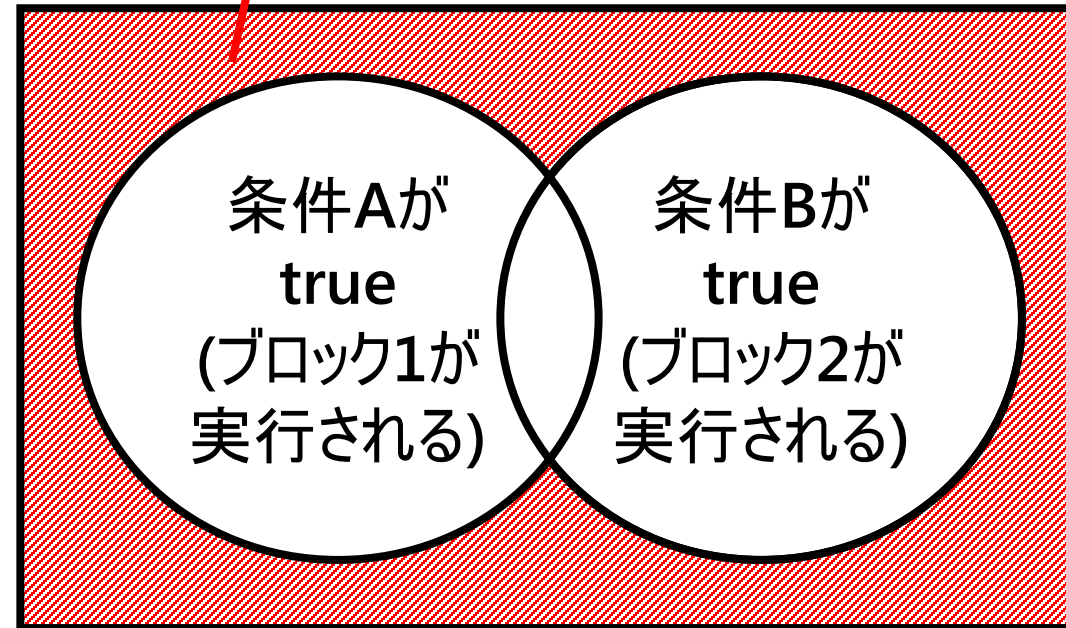
```
} else {
```

```
  文7;  
  文8;  
  文9;
```

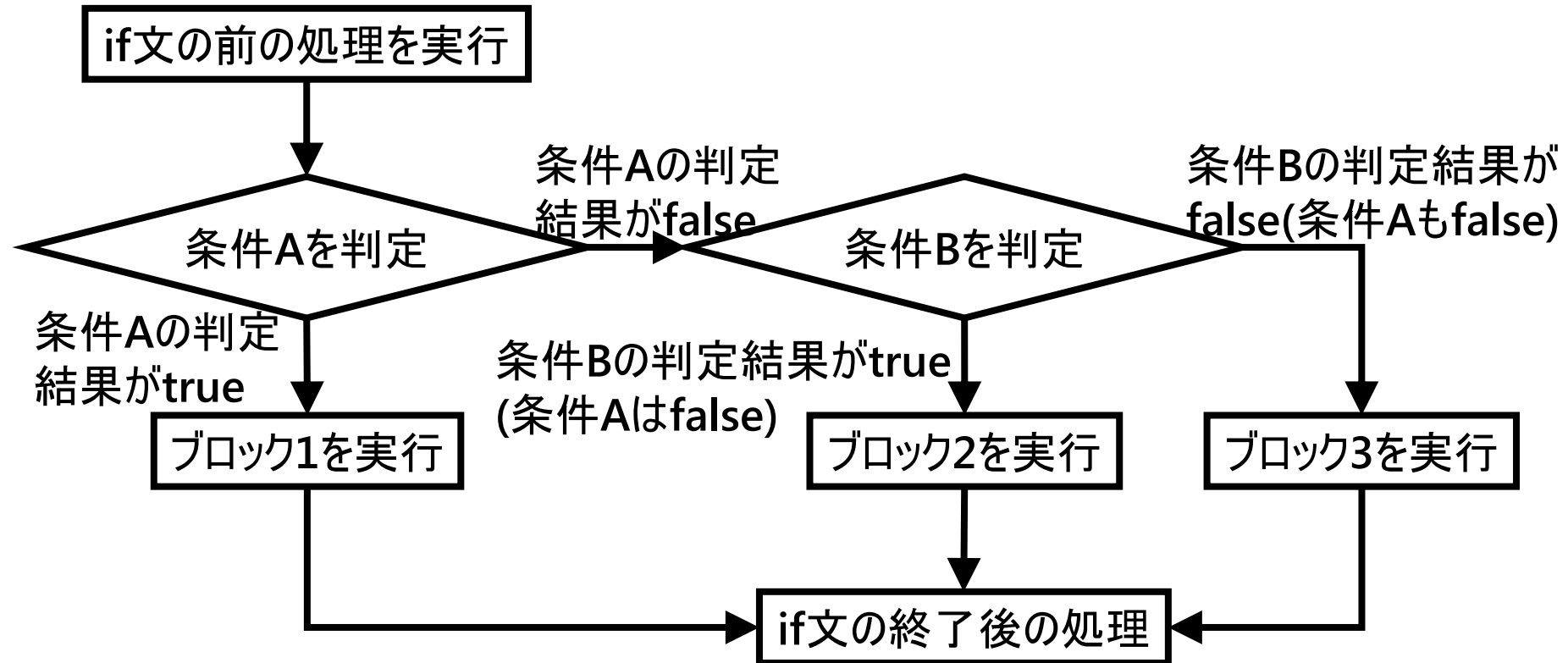
← ブロック3

```
}
```

「else」の部分(ブロック3)が
実行される範囲



条件分岐(発展)(3)(p. 156)





条件分岐(発展)(例)(p. 156)

例: レストランAがすいていればAで食事をし、AがすいていなくてBがすいていればBで食事をする。AもBもすいていなければ、あきらめて帰る



```
if (レストランAがすいている) {  
    レストランAで食事をする;  
} else if (レストランBがすいている) {  
    レストランBで食事をする;  
} else {  
    あきらめて帰る;  
}
```





「条件」のいろいろな書き方(p. 148)

⊗「条件A」が正しく、かつ「条件B」が正しい



(条件A) && (条件B)

if ((条件A) && (条件B))

⊗「条件A」が正しい、または「条件B」が正しい



(条件A) || (条件B)

if ((条件A) || (条件B))

⊗「条件A」が正しくない



!(条件A)

if (!(条件A))





「条件」のいろいろな書き方(例)(p. 148)

- ⌘ 服の値段(cloth)が1万円以下で、なおかつ靴の値段(shoes)が5千円以下

```
if ((cloth <= 10000) && (shoes <= 5000))
```

- ⌘ 服の値段(cloth)が1万円以下、または靴の値段(shoes)が5千円以下

```
if ((cloth <= 10000) || (shoes <= 5000))
```

- ⌘ 服(clothBrand)と靴(shoesBrand)のブランドが違う

```
if (!clothBrand.equals(shoesBrand))
```

条件をどう表現するか、あとは応用力!





複数の条件をつなげる

⌘ 演算子には、強弱の順序が決められている

⌘ Ex. 「 $5+8*10$ 」という計算では、「 $8*10$ 」をして、その結果に「5」を足し算する

Ex. if文の条件として...

`if (num1 > 10 && num2 < 20)`

➡ 「 $\text{num1} > 10$ 」を最初に判断し、次に「 $\text{num2} < 20$ 」を判断し、最後に2つの結果を「&&」で判断する

but...

正しく条件分岐をさせるには、演算子の強弱の順序を覚える必要

「()」を使って、どの式を先に判断させるか、明確にしておく方がベター
Ex. `if ((num1 > 10) && (num2 < 20))`



条件設定の注意事項(1)(p. 150)

⊗「**=**」は**代入**

⊗ Ex. $a = b$

⊗ この意味は、変数 a に b (値または変数) を代入する

⊗ この結果は、boolean 型ではない

⊗「**==**」は、**演算子の左右のものが同じかどうかを判定**

⊗ Ex. $a == b$

⊗ この意味は、 a (値または変数) と b (値または変数) が同じであるかどうかを判定する

⊗ この結果は、boolean 型で出る





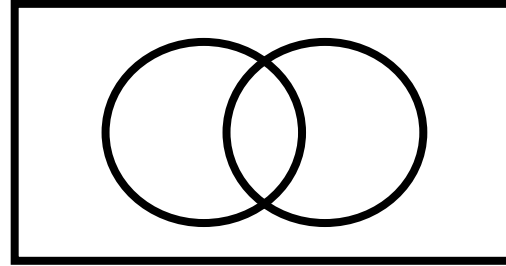
条件設定の注意事項(2)(p. 150)

- ⊗ 「==」で判定できるものとできないもの(ここまでの内容で)
 - ⊗ 数(int, float, double型)は「==」で判定可能
 - ⊗ 「num == 10」という書き方(numはint型の変数)はOK
 - ⊗ 文字列(String型)は「==」で判定不可能
 - ⊗ 「str == "abc"」という書き方(strはString型の変数)はNG
 - ⊗ String型は「equals」メソッドで判定(「if (str.equals("abc"))」のように書く)



条件設定の考え方

⚙️ 図を描いて考えてみる



⚙️ それぞれの条件にあてはまる実例を考えてみる

⚙️ 「else」になる条件を考えてみる

```
if ((num >= 0) && (num < 10)) {  
    [blue bar]  
} else if (num >= 10) {  
    [yellow bar]  
} else {  
    [grey bar]  
}
```

➤ numが10のときは?
➤ numが3のときは?
➤ numが-20のときは?



カッコのつけ方(p. 161)

※「{」から「}」までの間には、文をいくつ書いてもOK



```
if (条件) {  
    したいことその1;  
    したいことその2;  
    .....  
    したいことそのn;  
}
```

※「{」から「}」までの間に文が1つしかないときは、カッコを省略してもOK

※安全のために、常にカッコはつけておくべき



```
if (条件) したいこと;
```





「したいこと」の書き方

⊗「したいこと」の部分に書くものは、これまで書いてきたこと全て書くことができる

⊗計算

⊗標準入力

⊗標準出力

⊗文字列の操作

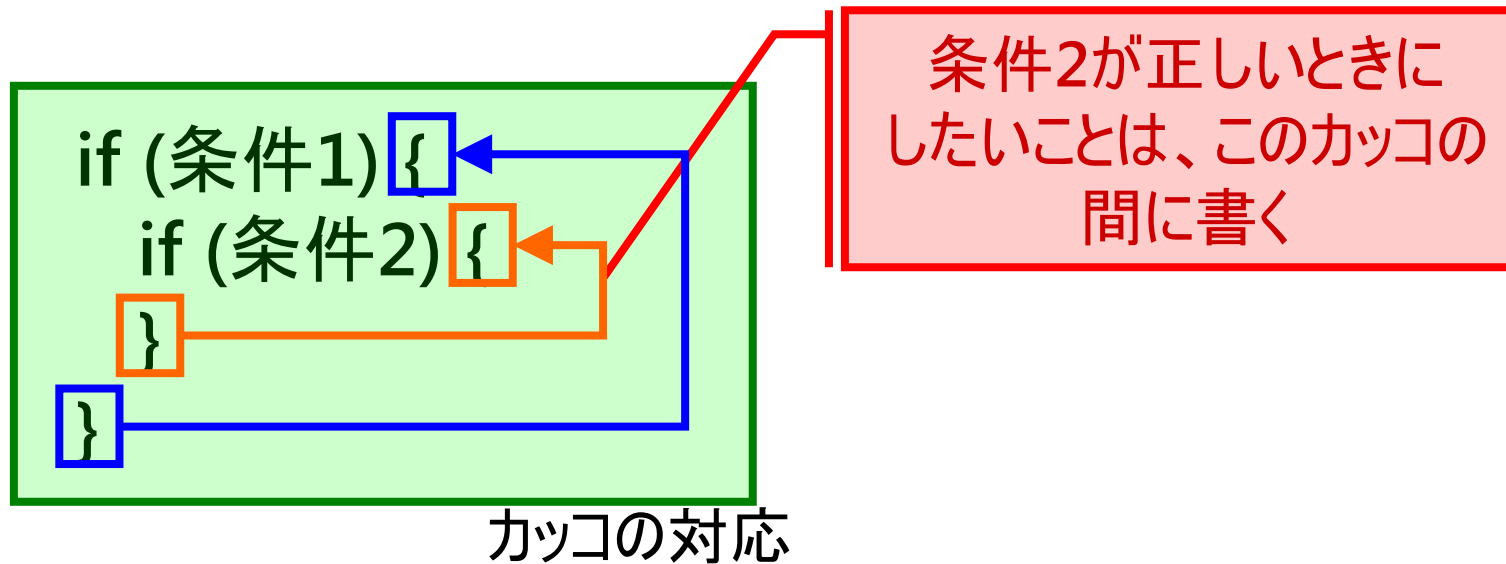
⊗etc.

⊗if文の「{ }」の中にさらにif文を書いてもOK



カッコの閉じ方

- ※ 開始カッコ1つに対して、閉じカッコが必ず1つ必要
- ※ カッコは内側から(閉じカッコは、一番近くの開始カッコを)閉じる



※ 開くカッコ「{」を書いたら、すぐに閉じるカッコ「}」を書いておくと、カッコを閉じ忘れない



プログラムでの書き方例(1)

⌘ 2つの数が同じ場合、「Equal」と出力し、同じでない場合には「Different」と出力するプログラム

```
if (num1 == num2) {  
    System.out.println("Equal");  
} else {  
    System.out.println("Different");  
}
```

※num1, num2: int型の変数





プログラムでの書き方例(2)

- ❁ 2つの文字列が同じ場合、「Equal」と出力し、同じでない場合には「Different」と出力するプログラム
- ❁ 文字列が同じであるかどうかは、「==」では比べられないので注意すること

「str1 == str2」という書き方は×

```
if (str1.equals(str2)) {  
    System.out.println("Equal");  
} else {  
    System.out.println("Different");  
}
```

※str1, str2はString型の変数





プログラムでの書き方例(3-1)

⊗ ある文字列中に、「,」が最初に出てくる位置(first)と最後(last)に出てくる位置を求め、

⊗ firstとlastが等しくなければ、first以前の部分文字列とlast以降の部分文字列を求める

⊗ firstとlastが等しければ、文字列の長さを求める

例えば、

- 「abc, def, ghi」の場合: 「abc」と「ghi」を求める
- 「abc, def」の場合: 文字列の長さ「7」を求める





プログラムでの書き方例(3-2)

```
first=str.indexOf(",");  
last=str.lastIndexOf(",");  
  
if (first != last) {  
    firstStr = str.substring(0, first);  
    lastStr = str.substring(last + 1);  
  
    System.out.println("First: " + firstStr);  
    System.out.println("Last: " + lastStr);  
} else {  
    System.out.println("String Length: " + str.length());  
}
```

※first, last: int型の変数

firstStr, lastStr: String型の変数





注意(1)

⚙ 変数の宣言は、if文の外ですること

```
int num;  
if (条件) {  
    num = 10;  
}
```

変数「num」は、if文の外でも
利用可能

```
if (条件) {  
    int num;  
    num = 10;  
}
```

変数「num」は、if文の中でのみ
利用可能

注意(2-1)(p. 159)

❁ if文の中で変数の初期化をする場合、必ず「else」も書いてその中でも初期化をすること

```
if (条件) {  
    num = 10;  
}  
result = num + 100;
```

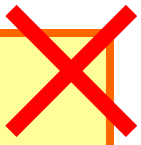
```
if (条件) {  
    num = 10;  
} else {  
    num = 20;  
}  
result = num + 100;
```

numはif文の条件が正しいときのみ初期化されるので、resultの計算のときにはnumに値が入っていない可能性
→コンパイルエラー

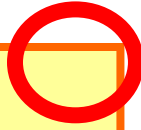
elseは「それ以外の場合」という意味なので、条件が正しくなかったときでもnumに値が入り、正しくコンパイル

注意(2-2)(p. 159)

❁ if文で「else」部分での初期化は、「else if」でなく「else」でも初期化をすること



```
if (i >= 0) {  
    num = 10;  
} else if (i < 0) {  
    num = 20;  
}  
result = num + 100;
```



```
if (i >= 0) {  
    num = 10;  
} else {  
    num = 20;  
}  
result = num + 100;
```

意味的な条件はまったく同じもの

but

コンピュータ的には違う条件と解釈

➤ コンピュータは、「 $i \geq 0$ 」でなく「 $i < 0$ 」でもない条件が存在する考える



やってみよう!

⌘教科書p. 173の例題01-06をやってみよう

