# プログラミング

- 第2回
  - if文（第4章）
  - switch文（第5章）
  - for文（第6章）
  - while文とString型（第7章）

# プログラミング

- 第2回
  - **if文（第4章）**
  - switch文（第5章）
  - for文（第6章）
  - while文とString型（第7章）

# if文(第4章)

- 条件分岐

```
if ( 条件式 ) {
        実行文 ;       // 複数可
}
```

～ C言語と同じ

# if文（第4章）

- 比較演算子

boolean型

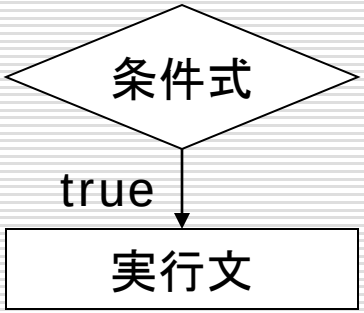
| 比較演算子 | 使用例 | 意味（true/false） |
|---|---|---|
| > | x > y | xがyよりも大きい場合true |
| >= | x >= y | xがyより大きいか等しい場合true |
| < | x < y | xがyより小さい場合true |
| <= | x <= y | xがyより小さいか等しい場合true |
| == | x == y | xとyが等しい場合true |
| != | x != y | xとyが等しくない場合true |

～ C言語と同じ

# if文（第4章）

## フローチャート

```
if ( 条件式 ) {
        実行文 ;
}
```




～ C言語と同じ

# if文(第4章)

- if … else …

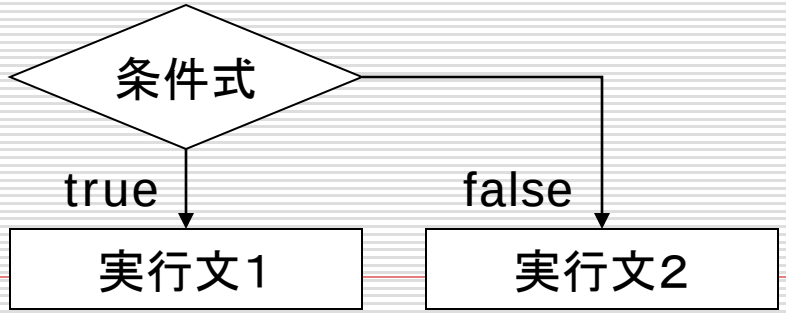
```
if ( 条件式 ) {
        実行文1 ;
} else {
        実行文2 ;
}
```

～ C言語と同じ

# if文（第4章）

## フローチャート

```
if ( 条件式 ) {
        実行文1 ;
} else {
        実行文2 ;
}
```




～ C言語と同じ

# if文（第4章）

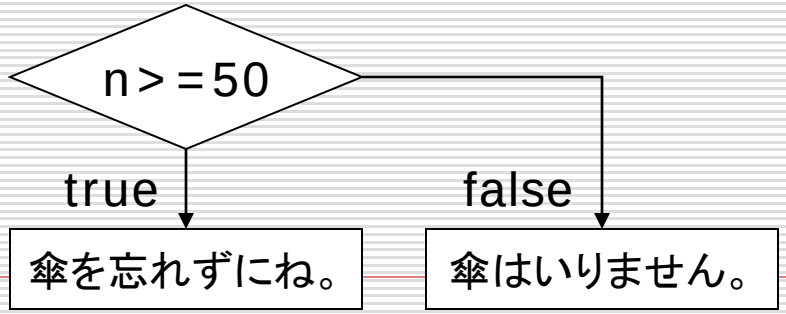
## 例題1 (Kasa1.java)

```
import java.io.*;

public class Kasa1 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         System.out.println("降水確率を入力してください。");
         String line = reader.readLine();
         int n = Integer.parseInt(line);
         System.out.println("降水確率は" + n + "%です。");
         if (n >= 50) {
            System.out.println("傘を忘れずにね。");
         } else {
            System.out.println("傘はいりません。");
         }
         System.out.println("いってらっしゃい。");
      } catch (IOException e) {
         System.out.println(e);
      } catch (NumberFormatException e) {
         System.out.println("数字の形式が正しくありません。");
      }
   }
}
```

# if文（第4章）

## フローチャート

```
if (n >= 50) {
        System.out.println("傘を忘れずにね。");
} else {
        System.out.println("傘はいりません。");
}
```



# if文(第4章)

## 例題2 (Kasa2.java)

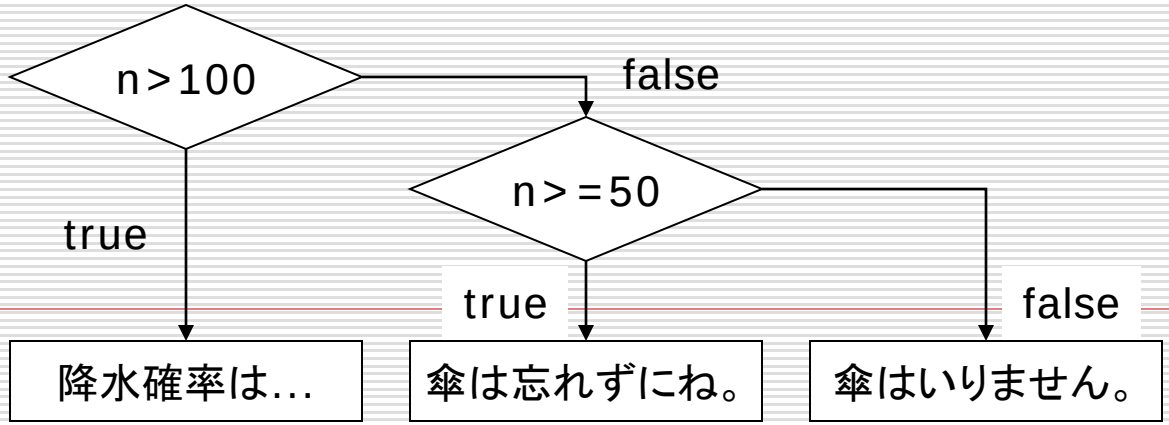
```
import java.io.*;

public class Kasa2 {
    public static void main(String[] args) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            System.out.println("降水確率を入力してください。");
            String line = reader.readLine();
            int n = Integer.parseInt(line);
            System.out.println("降水確率は" + n + "%です。");
            if (n > 100) {
                System.out.println("降水確率は 0 ～ 100 の間ですよ。");
            } else if (n >= 50) {
                System.out.println("傘を忘れずにね。");
            } else {
                System.out.println("傘はいりません。");
            }
            System.out.println("いってらっしゃい。");
        } catch (IOException e) {
            System.out.println(e);
        } catch (NumberFormatException e) {
            System.out.println("数字の形式が正しくありません。");
        }
    }
}
```

# if文（第4章）

## □ if文の連鎖 (else if)

エラーメッセージ

```
if (n > 100) {
        System.out.println("降水確率は 0 ～ 100 の間ですよ。");
} else if (n >= 50) {
        System.out.println("傘を忘れずにね。");
} else {
        System.out.println("傘はいりません。");
}
```



# if文(第4章)

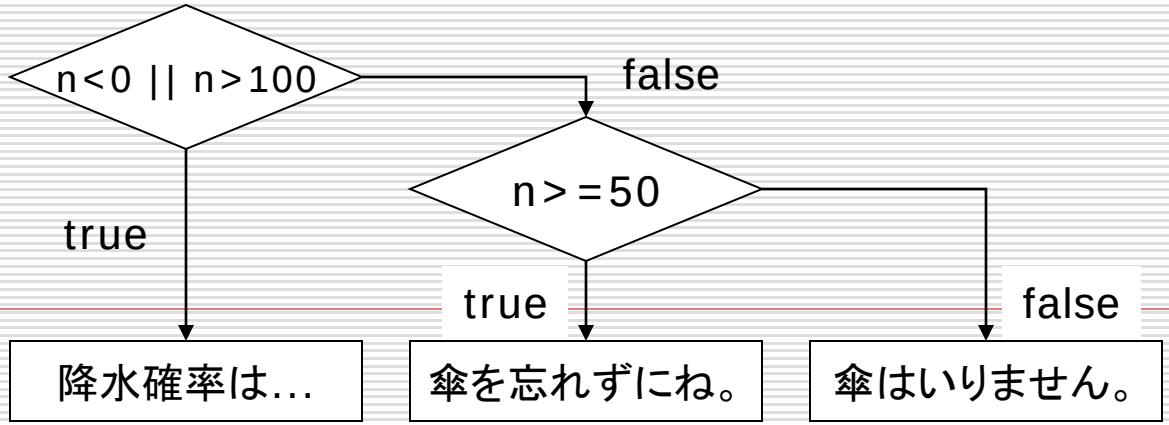
## 例題3 (Kasa3.java)

```
import java.io.*;

public class Kasa3 {
    public static void main(String[] args) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            System.out.println("降水確率を入力してください。");
            String line = reader.readLine();
            int n = Integer.parseInt(line);
            System.out.println("降水確率は" + n + "%です。");
            if (n < 0 || 100 < n) {
                System.out.println("降水確率は 0 ～ 100 の間ですよ。");
            } else if (n >= 50) {
                System.out.println("傘を忘れずにね。");
            } else {
                System.out.println("傘はいりません。");
            }
            System.out.println("いってらっしゃい。");
        } catch (IOException e) {
            System.out.println(e);
        } catch (NumberFormatException e) {
            System.out.println("数字の形式が正しくありません。");
        }
    }
}
```

# if文（第4章）

## 論理和 ||

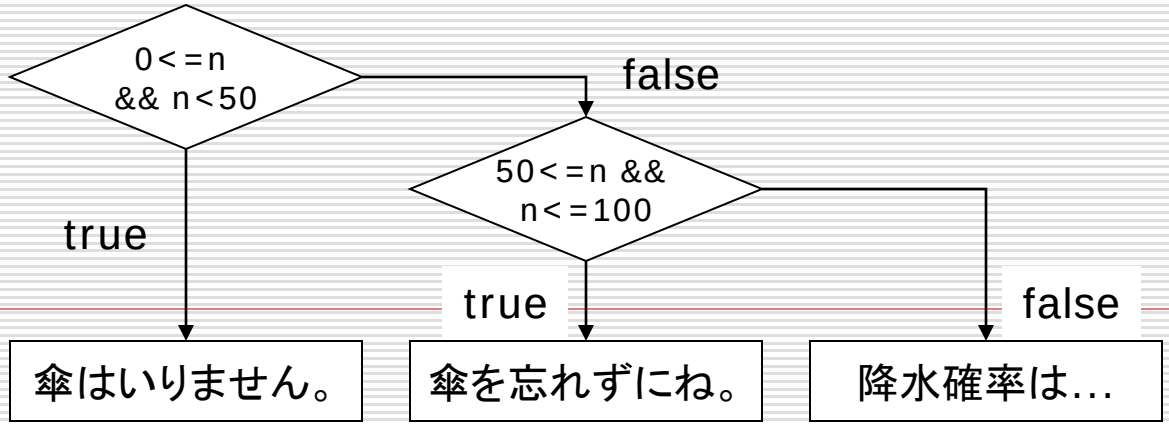
```
if (n < 0 || 100 < n) {
        System.out.println("降水確率は 0 ～ 100 の間ですよ。");
} else if (n >= 50) {
        System.out.println("傘を忘れずにね。");
} else {
        System.out.println("傘はいりません。");
}
```



# if文（第4章）

## 論理積 &&

```
if (0 <= n && n < 50) {
        System.out.println(“傘はいりません。");
} else if (50 <= n && n <= 100) {
        System.out.println(“傘を忘れずにね。");
} else {
        System.out.println("降水確率は 0 ～ 100 の間ですよ。");
}
```



# if文（第4章）

- 論理演算子

| 論理演算子 | 使用例 | 説明 |
|---|---|---|
| && | a && b | aとbの両方がtrueの場合にtrue |
| \|\| | a \|\| b | aとbのいずれかがtrueの場合にtrue |
| ! | !a | aがfalseの場合にtrue |

～ C言語と同じ

# プログラミング

- 第2回
  - if文（第4章）
  - **switch文（第5章）**
  - for文（第6章）
  - while文とString型（第7章）

# switch文（第5章）

- if文による多方向分岐

```
if ( n == 1 ) {
        実行文1 ;
} else if ( n == 2 ) {
        実行文2 ;
} else {
        実行文3 ;
}
```

～ C言語と同じ

# switch文（第5章）

- switch文による多方向分岐

```
switch ( n ) {
case 1 :
      実行文1 ;
      break ;
case 2 :
      実行文2 ;
      break ;
default :
      実行文3 ;
      break ;
}
```

～ C言語と同じ

# switch文(第5章)

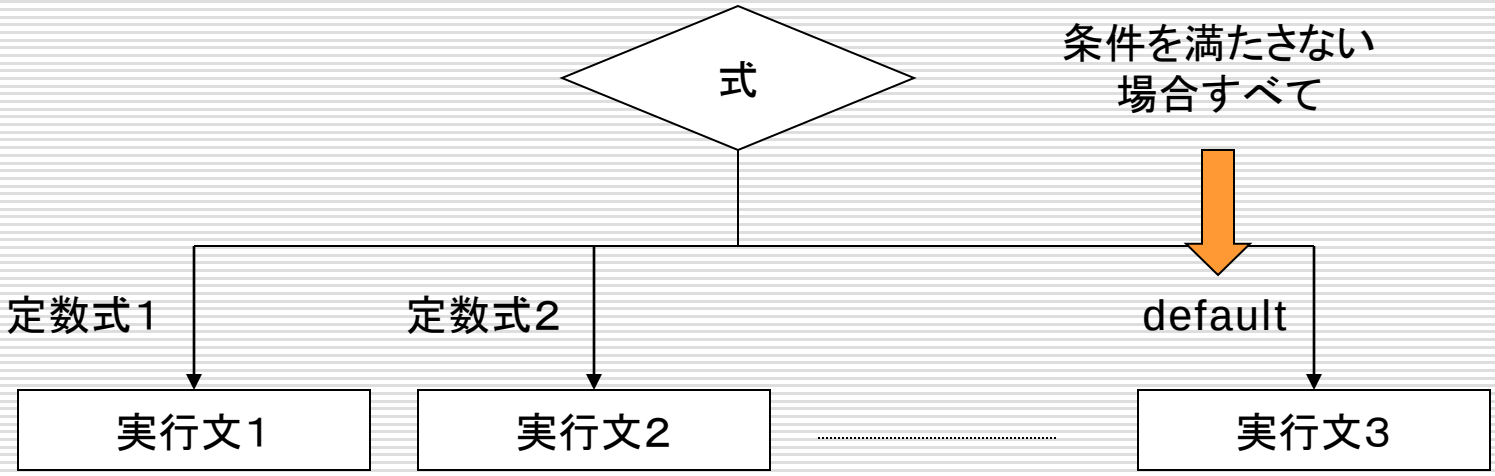
## switch文の構文

```
switch ( 式 ) {
case 定数式1 :
        実行文1 ;
        break ;
case 定数式2 :
        実行文2 ;
        break ;
default :
        実行文3 ;
        break ;
}
```

～ C言語と同じ

# switch文（第5章）

## フローチャート




～ C言語と同じ

# switch文（第5章）

## 例題1 (Drink1.java)

```
import java.io.*;

public class Drink1 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         System.out.println("飲み物は何が好きですか？");
         System.out.println("1 オレンジジュース");
         System.out.println("2 コーヒー");
         System.out.println("3 どちらでもない");
         System.out.println("1,2,3のどれかを選んでください。");
         String line = reader.readLine();
         int n = Integer.parseInt(line);
         switch (n) {
         case 1:
            System.out.println("オレンジジュースです。");
            break;
         case 2:
            System.out.println("コーヒーです。");
            break;
         default:
            System.out.println("どちらでもありません。");
            break;
         }
      } catch (IOException e) {
         System.out.println(e);
      } catch (NumberFormatException e) {
         System.out.println("数字の形式が正しくありません。");
      }
   }
}
```

数字以外が入力された場合 ⇒ 例外発生

# switch文（第5章）

- 例題1抜粋

```
switch (n) {
case 1:
        System.out.println("オレンジジュースです。");
        break;
case 2:
        System.out.println("コーヒーです。");
        break;
default:
        System.out.println("どちらでもありません。");
        break;
}
```

# switch文（第5章）

## 例題1 (Drink1.java)

```
import java.io.*;

public class Drink2 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         System.out.println("飲み物は何が好きですか？");
         System.out.println("a オレンジジュース");
         System.out.println("b コーヒー");
         System.out.println("c どちらでもない");
         System.out.println("a,b,cのどれかを選んでください。");
         String line = reader.readLine();
         char c = line.charAt(0);
         switch (c) {
         case 'a':
            System.out.println("オレンジジュースです。");
            break;
         case 'b':
            System.out.println("コーヒーです。");
            break;
         default:
            System.out.println("どちらでもありません。");
            break;
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# switch文（第5章）

- 文字による分岐

```
switch (c) {
case 'a':  ⇐ 文字定数
        System.out.println("オレンジジュースです。");
        break;
case 'b':
        System.out.println("コーヒーです。");
        break;
default:
        System.out.println("どちらでもありません。");
        break;
}
```

# switch文（第5章）

## Stringクラス、charAtメソッド

```
System.out.println("飲み物は何が好きですか？");
System.out.println("a オレンジジュース");
System.out.println("b コーヒー");
System.out.println("c どちらでもない");
System.out.println("a,b,cのどれかを選んでください。");
String line = reader.readLine();
char c = line.charAt(0);                    // 0は先頭位置
```

- 文字列中の指定された位置の文字を返すメソッド

| | |
|---|---|
| “abc” | 文字列（改行コード含む） |
| “a” | 文字列（改行コード含む） |
| ‘a’ | 文字 |

java.lang.String

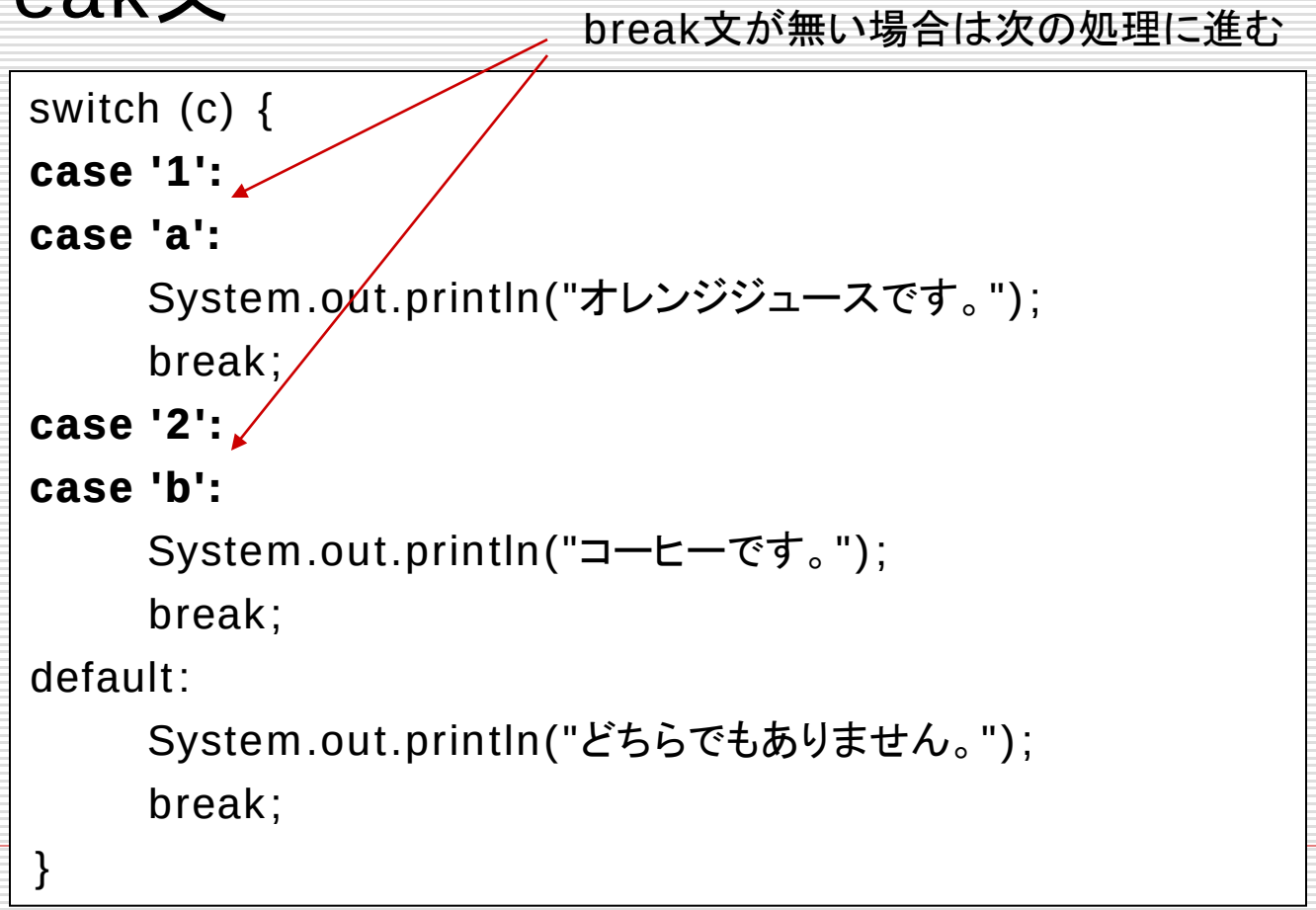
# switch文（第5章）

## 例題3 (Drink3.java)

```
import java.io.*;

public class Drink3 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         System.out.println("飲み物は何が好きですか？");
         System.out.println("1 オレンジジュース(a)");
         System.out.println("2 コーヒー(b)");
         System.out.println("3 どちらでもない(c)");
         System.out.println("1,2,3のどれかを選んでください（a,b,cでも選べます）。");
         String line = reader.readLine();
         char c = line.charAt(0);
         switch (c) {
         case '1':
         case 'a':
            System.out.println("オレンジジュースです。");
            break;
         case '2':
         case 'b':
            System.out.println("コーヒーです。");
            break;
         default:
            System.out.println("どちらでもありません。");
            break;
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# switch文（第5章）

- break文



# switch文（第5章）

- 定数式

```
switch ( 式 ) {
case 定数式1 :
      実行文1 ;
      break ;
case 定数式2 :
      実行文2 ;
      break ;
default :
      実行文3 ;
      break ;
}
```

- switch文の定数式は整数に限定される
- 文字も整数
- long型、double型、String型などは不可

# プログラミング

- 第2回
  - if文（第4章）
  - switch文（第5章）
  - **for文（第6章）**
  - while文とString型（第7章）

# for文（第6章）

- 繰り返し (Count2.java)

```
public class Count2 {
    public static void main(String[] args) {
        for (int i = 0; i < 3; i++) {
            System.out.println(i);
        }
        System.out.println("end");
    }
}
```

i = 0, 1, 2

～ ほぼC言語と同じ

# for文(第6章)

- 構文

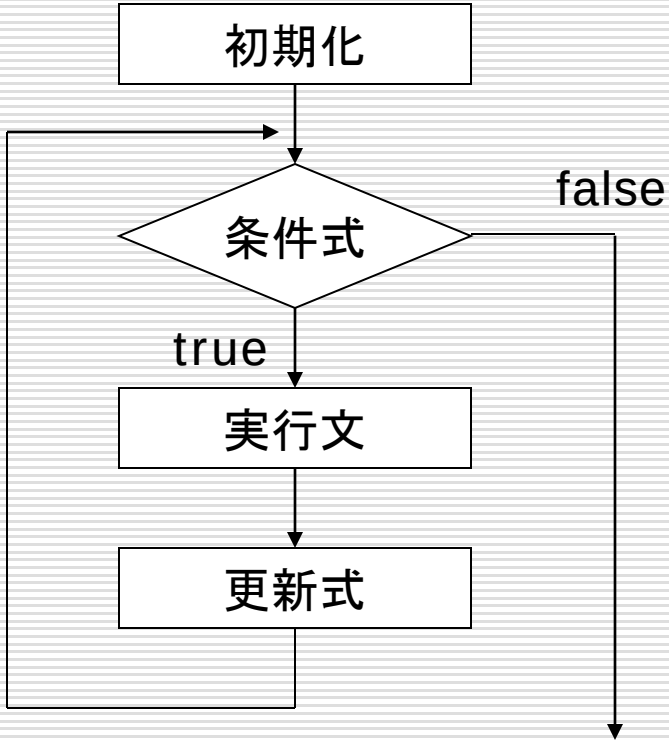
```
for ( 初期化 ; 条件式 ; 更新式 ) {
        実行文 ;
}
```

～ C言語と同じ

# for文（第6章）

## □ フローチャート

```
for ( 初期化 ; 条件式 ; 更新式 ) {
        実行文 ;
}
```




～ C言語と同じ

# for文（第6章）

## □ 変数の宣言とスコープ

型宣言：C言語ではコンパイルエラー

```
public class Count2 {
    public static void main(String[] args) {
        for (int i = 0; i < 3; i++) {
            System.out.println(i);
        }
        System.out.println("end");
    }
}
```

変数 i のスコープ
（有効範囲）

# for文（第6章）

- インクリメント / デクリメント

| 名称 | 演算 | 表記 | 意味 |
|---|---|---|---|
| インクリメント | y = y − 1; | y++ ; | 評価してからy+1に更新 |
| | | ++y ; | y+1に更新してから演算 |
| デクリメント | y = y − 1; | y−− ; | 評価してからy−1に更新 |
| | | −−y ; | y−1に更新してから演算 |

～ C言語と同じ

# for文（第6章）

- 例題3 (Count3.java)

```
public class Count3 {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.print(i + "の2乗は" + (i * i) + "で、");
            System.out.println("3乗は" + (i * i * i) + "です。");
        }
        System.out.println("end");
    }
}
```

# for文（第6章）

- ディスプレイへの表示

```
for (int i = 0; i < 10; i++) {
    System.out.print(i + "の2乗は" + (i * i) + "で、");
    System.out.println("3乗は" + (i * i * i) + "です。");
}
System.out.println("end");
```

- System.out.print：改行なし
- System.out.println：改行あり

# for文（第6章）

- 例題4 (DrawGraph1.java)

```
public class DrawGraph1 {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.print(i + ":");
            for (int j = 0; j < i; j++) {
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

i = 0, 1, 2, ... , 9

j = 0, 1, ... , i-1

二重のfor文 ～ for文の「入れ子」

# for文（第6章）

- { ... }　中括弧

```
for (int i = 0; i < 10; i++) {
    System.out.println (i) ;
}
```

=

```
for (int i = 0; i < 10; i++)
    System.out.println (i) ;
```

- if文やwhile文の場合も省略可能
- ただし、実行文が一つだけの場合に限る

# for文（第6章）

- 変数のスコープ (再)

```
public class CountE {
    public static void main(String[] args) {
        for (int i = 0; i < 3; i++) {
            System.out.println(i);
        }
        System.out.println("i = " + i);
        System.out.println("end");
    }
}
```

変数 i のスコープ（有効範囲）

コンパイルエラー

# for文（第6章）

- 変数のスコープ (再)

```
public class CountOk {
    public static void main(String[] args) {
        int i;          // for文の外で変数宣言
        for (i = 0; i < 3; i++) {
            System.out.println(i);
        }
        System.out.println("i = " + i);
        System.out.println("end");
    }
}
```

変数 i のスコープ（有効範囲）

コンパイルエラーなし

# プログラミング

- 第2回
  - if文（第4章）
  - switch文（第5章）
  - for文（第6章）
  - **while文とString型（第7章）**

# while文とString型（第7章）

- 繰り返し

```
while ( 条件式 ) {
        実行文 ;
}
```

～ C言語と同じ

# while文とString型（第7章）

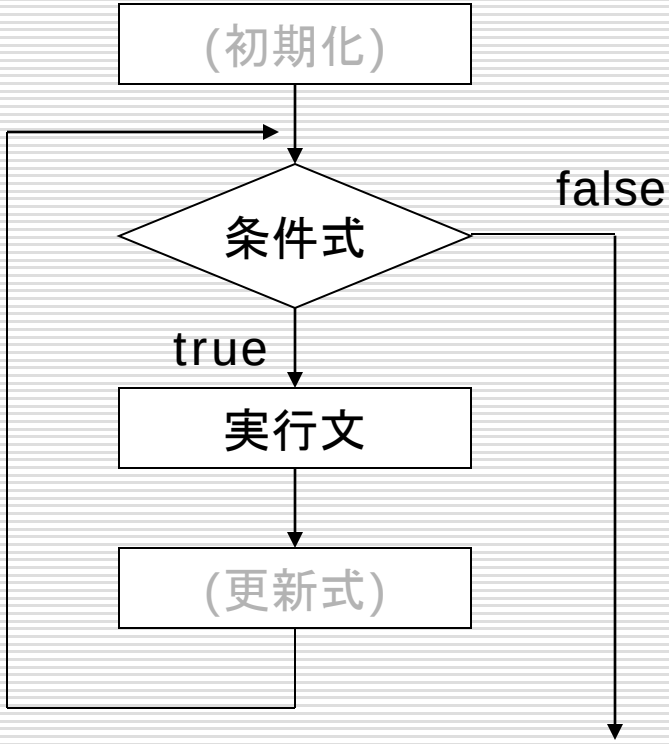
- 繰り返し (While1.java)

```
public class While1 {
    public static void main(String[] args) {
        int i = 0;              // 変数宣言＆初期化
        while (i < 3) {
            System.out.println(i);
            i++;
        }
        System.out.println("end");
    }
}
```

i = 0, 1, 2

# while文とString型（第7章）

## フローチャート




```
（初期化）
while ( 条件式 ) {
        実行文 ;
        （更新式）
}
```

～ C言語と同じ

# while文とString型（第7章）

- 例題 (Copy1.java)

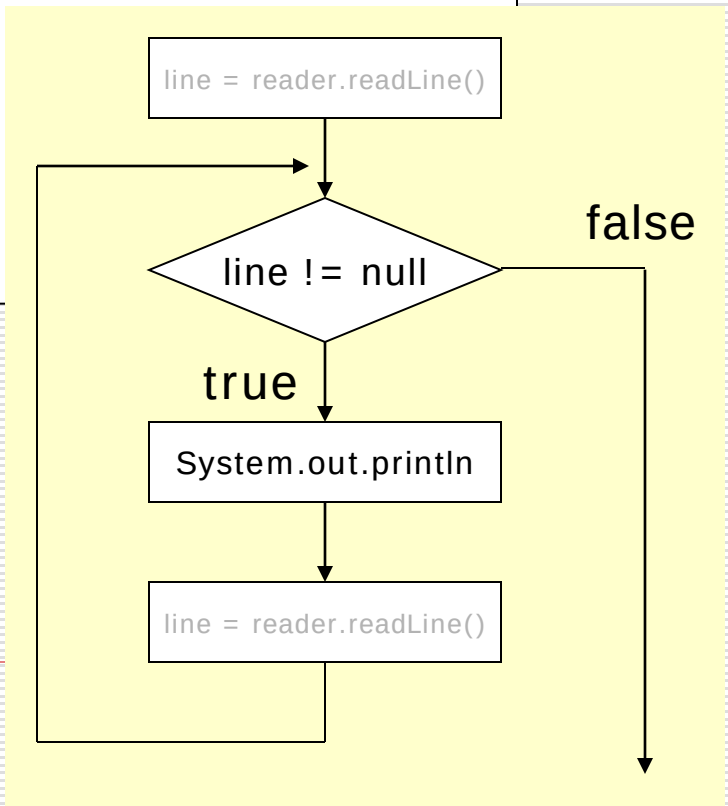
```
import java.io.*;

public class Copy1 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         String line = reader.readLine();
         while (line != null) {
            System.out.println(line);
            line = reader.readLine();
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# while文とString型（第7章）

- 例題 (Copy1.java)

```
String line = reader.readLine();
while (line != null) {
        System.out.println(line);
        line = reader.readLine();
}
```

Ctrl-z で終了
Ctrl-c で強制終了



# while文とString型（第7章）

- リダイレクション
  - < 標準入力をキーボードからファイルに切り替え
    - ファイル input.txt を読み込んで表示

```
C:¥Java> java Copy1 < input.txt
```

  - > 標準出力をディスプレイからファイルに切り替え
    - ファイル input.txt を読み、ファイルoutput.txt に書き込み

```
C:¥Java> java Copy1 < input.txt > output.txt
```

# while文とString型（第7章）

- 例題 (Copy2.java)

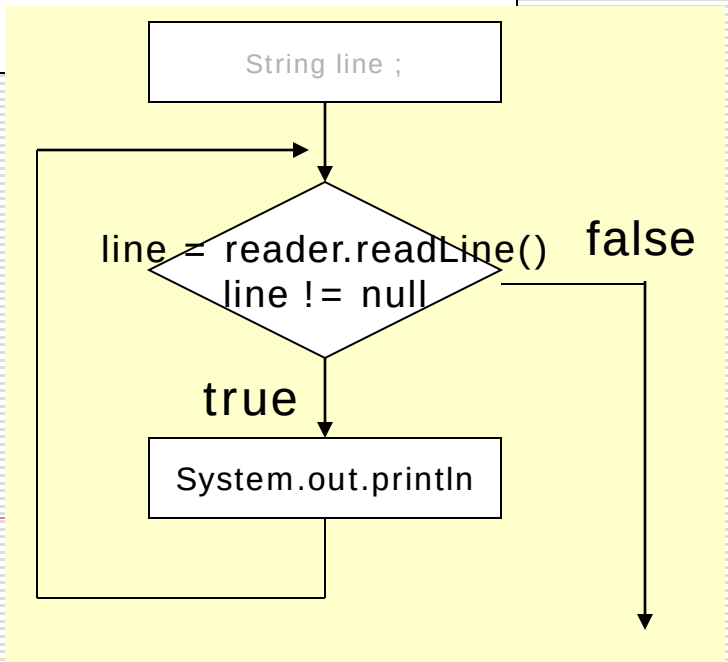
```
import java.io.*;

public class Copy2 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         String line;
         while ((line = reader.readLine()) != null) {
            System.out.println(line);
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# while文とString型（第7章）

- 例題 (Copy2.java)

Ctrl-z で終了
Ctrl-c で強制終了

```
String line;
while ((line = reader.readLine()) != null) {
        System.out.println(line);
}
```




- 動作は Copy1.java と同じ

# while文とString型（第7章）

- 例題 (CopyLower.java)

```
import java.io.*;

public class CopyLower {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         String line;
         while ((line = reader.readLine()) != null) {
            String s = line.toLowerCase();
            System.out.println(s);
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# while文とString型（第7章）

- Stringクラス、toLowerCaseメソッド

```
String line;
while ((line = reader.readLine()) != null) {
        String s = line.toLowerCase();
        System.out.println(s);
}
```

  - 大文字を小文字に変換するメソッド
    - A-Z ⇒ a-z

# while文とString型（第7章）

- 例題 (Convert1.java)

```
import java.io.*;

public class Convert1 {
   public static void main(String[] args) {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         String line;
         while ((line = reader.readLine()) != null) {
            String s = line.replace('。', '.');
            s = s.replace('、', ',');
            System.out.println(s);
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# while文とString型（第7章）

- Stringクラス、replaceメソッド

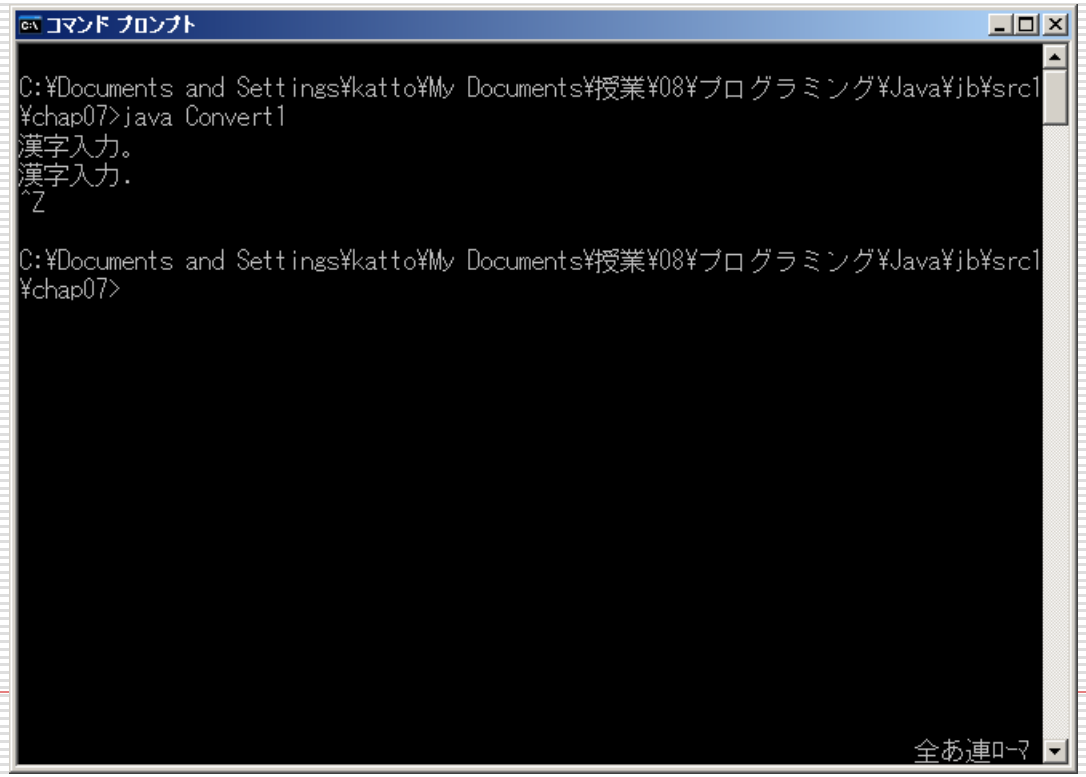
```
String line;
while ((line = reader.readLine()) != null) {
        String s = line.replace('。', '.');
        s = s.replace('、', ',');
        System.out.println(s);
}
```

。 ⇒ .
、 ⇒ ,

- 特定の部分文字列を置換するメソッド

# while文とString型（第7章）

- コマンドプロンプトでの漢字入力
  - Alt ＋ 漢字



# while文とString型（第7章）

- Stringクラスのメソッド
  - replace　　：置換
  - substring　　：部分文字列
  - toLowerCase　　：大文字⇒小文字
  - toString　　：オブジェクト（文字列）自身
  - toUpperCase　　：小文字⇒大文字
  - trim　　：文字列コピー
  - …
  - *see* Java API リファレンス

java.lang.String

# while文とString型（第7章）

- 以降は少し発展的な話：
  - 参照型変数
  - ガーベッジコレクション
  - 配列
  - 多重定義（オーバーロード）
  - …

# while文とString型（第7章）

## Stringオブジェクトと変数の関係

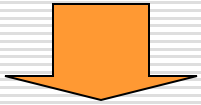
```
line = reader.readLine();
String s = line.replace('。', '.');
s = s.replace('、', ',');
System.out.println(s);
```

# while文とString型（第7章）

- メモリ消費とガーベッジコレクション

```
while( (line = reader.readLine()) != null) {
        String s = line.replace('。', '.');
        s = s.replace('、', ',');
        System.out.println(s);
}
```

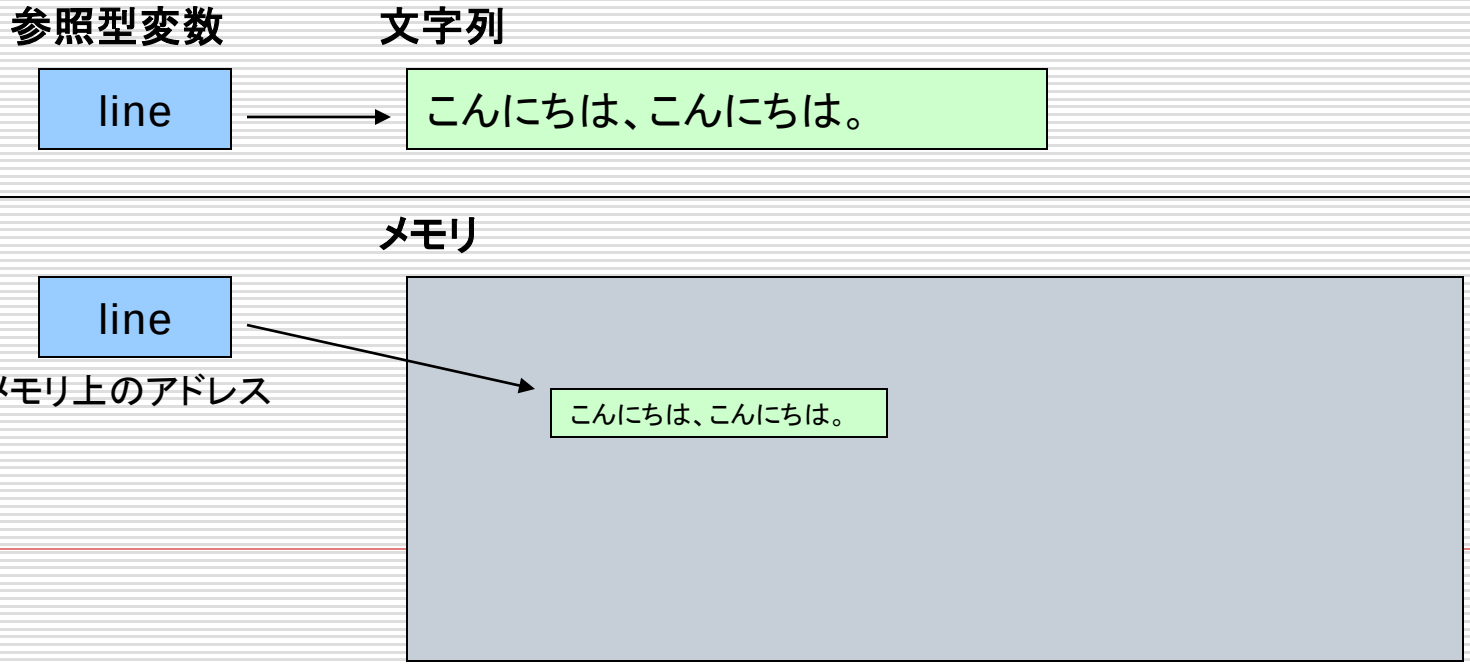
- ループのたびに String オブジェクトを3個作成
- そのたびにメモリを消費



- ガーベッジコレクション：使われなくなったオブジェクト（ガーベッジ＝ごみ）を集めてメモリを解放（第15章）

# while文とString型（第7章）

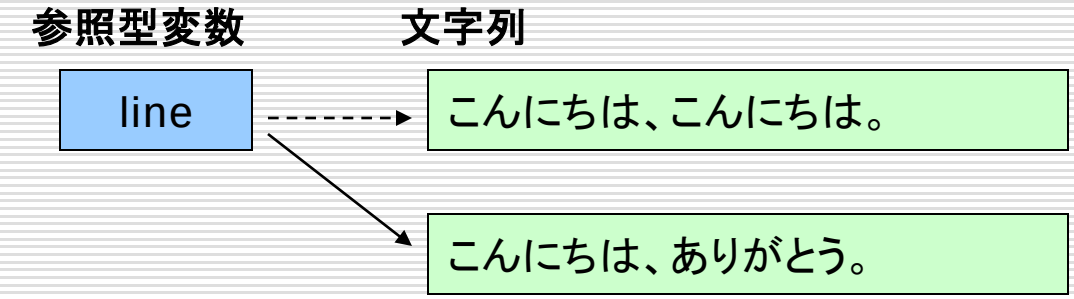
- 「参照」
  - 基本型: int, char, double, …
  - 参照型: String, 配列, …

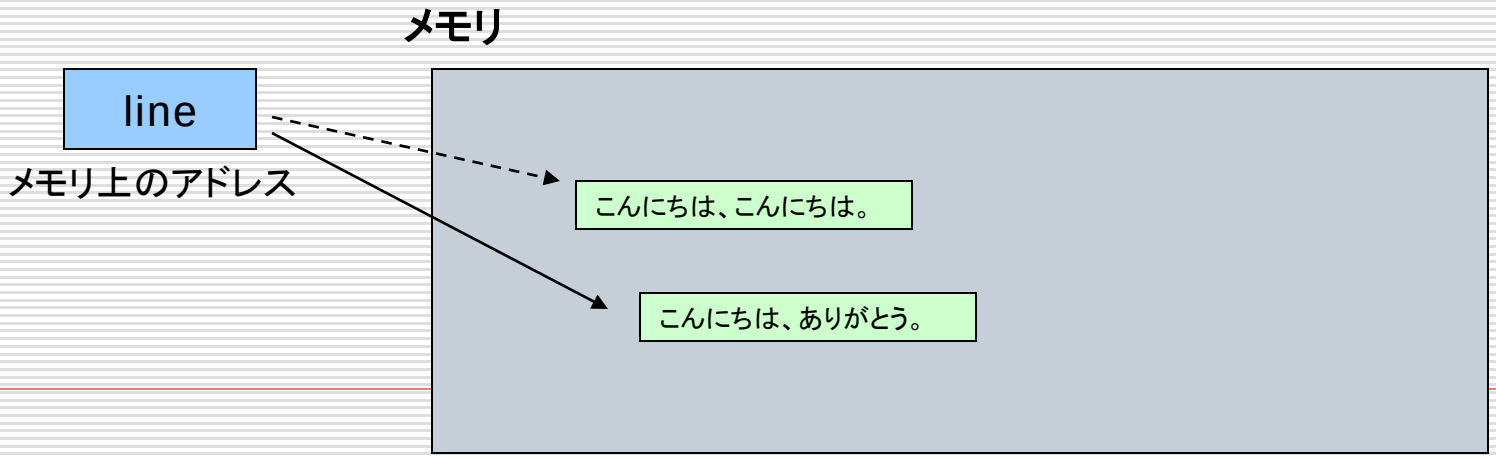


# while文とString型（第7章）

## □「参照」

String line = “こんにちは、こんにちは。”;
line = “こんにちは、ありがとう。”;

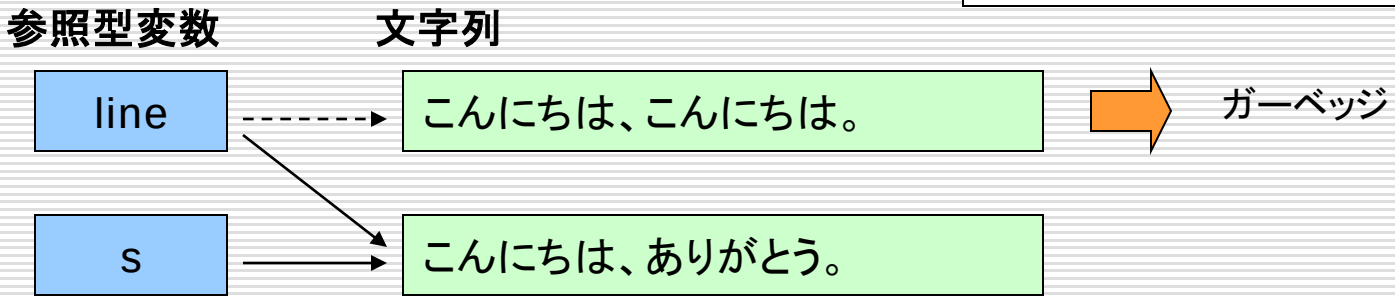


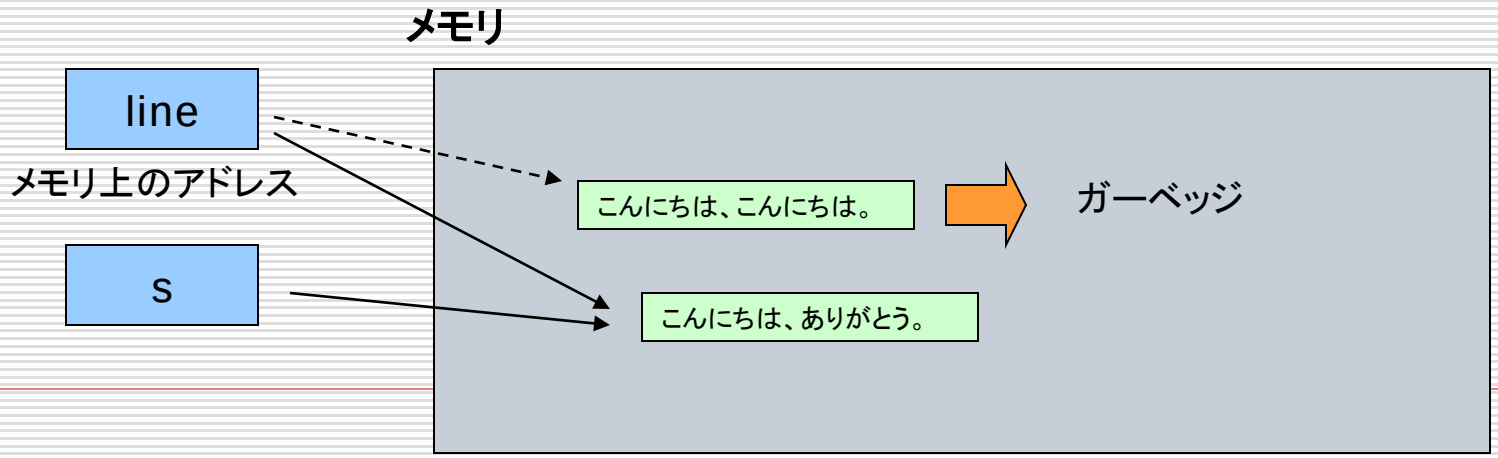



# while文とString型（第7章）

## □「参照」

```
String line = “こんにちは、こんにちは。”;
line = “こんにちは、ありがとう。”;
String s = line ;
```

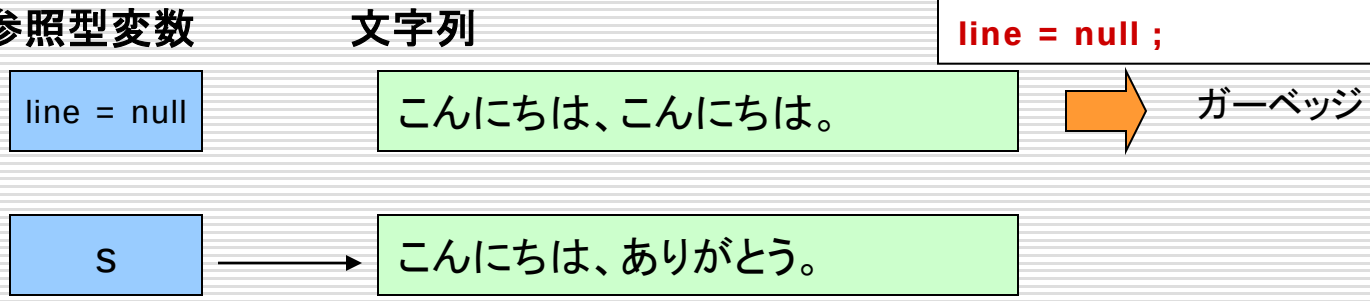


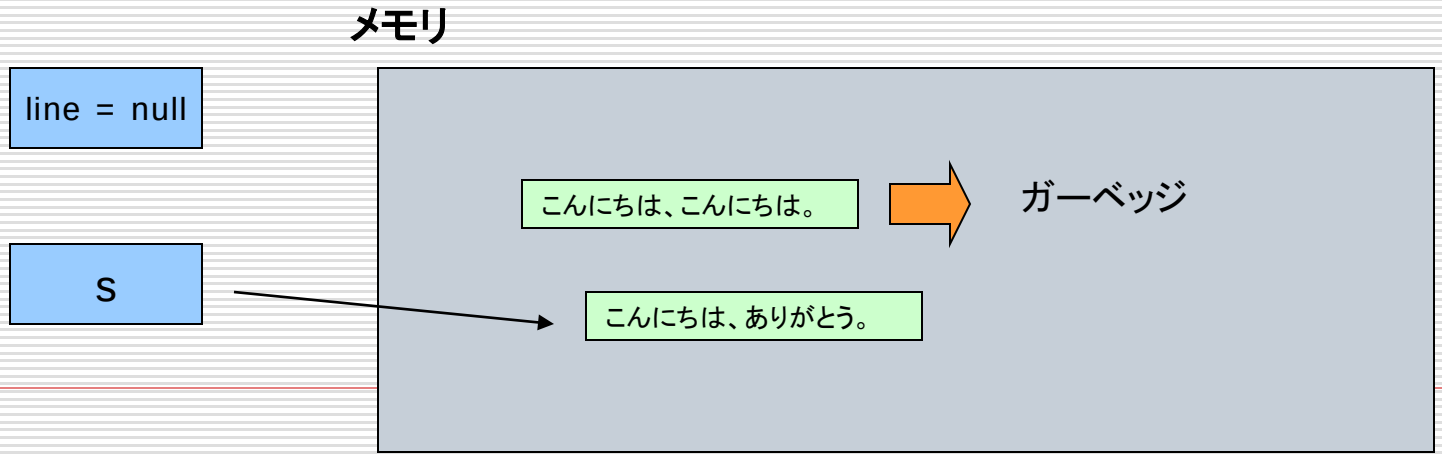



# while文とString型（第7章）

## □「参照」

```
String line = “こんにちは、こんにちは。”;
line = “こんにちは、ありがとう。”;
String s = line ;
line = null ;
```



# while文とString型（第7章）

- 「参照」
  - メモリ上のアドレス
  - C言語のポインタに相当

  - “Javaにはポインタがない”
  - “Java Technology: The Early Years”
    - C/C++とJavaの最大の違いは、Javaには、メモリへの上書きおよびデータ喪失の可能性を排除するポインタモデルがあることである。そこでは、ポインタ演算の替りに真の配列が用意されており、添字チェックを可能としている。なお、任意の整数をポインタに型変換することは、不可能である。

# while文とString型（第7章）

- 例題 (Find1.java)

```
import java.io.*;

public class Find1 {
   public static void main(String[] args) {
      if (args.length != 1) {
         System.out.println("使用法:java Find1 検索文字列 < 検索対象ファイル");
         System.out.println("例:java Find1 System < Find1.java");
         System.exit(0);
      }
      String findstring = args[0];
      System.out.println("検索文字列は「" + findstring + "」です。");
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      try {
         String line;
         int linenum = 1;
         while ((line = reader.readLine()) != null) {
            int n = line.indexOf(findstring);
            if (n >= 0) {
               System.out.println(linenum + ":" + line);
            }
            linenum++;
         }
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

# while文とString型（第7章）

## コマンドライン入力

```
public static void main(String[] args) {
    if (args.length != 1) {
        System.out.println("...");
        System.exit(0);
    }
    String findstring = args[0];
    ...
}
```

- args：コマンドライン入力（引数）の文字列配列
- args.length：引数の個数
- args[0]：最初の引数

# while文とString型（第7章）

- 使用例
  - java Find1 System < Find1.java
    - args.length = 1 (“System” のみ)
    - args[0] = “System”
    - < … リダイレクション（ファイル入力）
    - Find1.java ファイル内の文字列 “System” を検索

# while文とString型（第7章）

- Stringクラスと配列
  - Stringクラスのlengthはメソッド
  - 配列のlengthはフィールド

```
public static void main(String[] args) {
    if ( args.length != 1)
        System.exit(0);
    String findstring = args[0];
    System.out.println( findstring.length() );
}
```

# while文とString型（第7章）

- String クラスの indexOf メソッド

```
String line;
int linenum = 1;
while ((line = reader.readLine()) != null) {
    int n = line.indexOf(findstring);
    if (n >= 0) {
        System.out.println(linenum + ":" + line);
    }
    linenum++;
}
```

- 指定した文字列が含まれているかを調べる
- 含まれる場合は先頭の文字位置を返す(0以上)
- 含まれない場合は -1 を返す

# while文とString型（第7章）

- 多重定義（オーバーロード）
  - 複数のメソッドが同じ名前を使うこと
  - 引数の型と個数：シグニチャ
  - 例：String クラスの indexOf メソッド

| 戻り値 | メソッド概要 |
|---|---|
| int | **indexOf**(int ch)<br>この文字列内で、指定された文字が最初に出現する位置のインデックスを返します。 |
| int | **indexOf**(int ch, int fromIndex)<br>この文字列内で、指定されたインデックスから検索を開始し、指定された文字が最初に出現する位置のインデックスを返します。 |
| int | **indexOf**(String str)<br>この文字列内で、指定された部分文字列が最初に出現する位置のインデックスを返します。 |
| int | **indexOf**(String str, int fromIndex)<br>指定されたインデックス以降で、指定された部分文字列がこの文字列内で最初に出現する位置のインデックスを返します。 |

# while文とString型（第7章）

- Stringクラスのメソッド II（文字列検索）
  - charAt　　：指定位置の文字
  - equals　　：文字列の比較
  - compareTo　　：文字列の比較
  - regionMatches　：文字領域の比較
  - startsWith　　：接頭辞の比較
  - indexOf　　：文字列の位置（先頭）
  - lastIndexOf　　：文字列の位置（最後）
  - …
  - *see* Java API リファレンス

# while文とString型（第7章）

- 比較
  - 基本型：比較演算子
    - x == y
  - 参照型：equalsメソッド
    - x.equals(y)

# while文とString型（第7章）

- do-while文

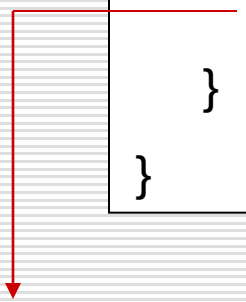
```
do {
        実行文；
} while（条件式）；
```

～ C言語と同じ

# while文とString型（第7章）

- break文
  - 繰り返しの中断

```
while ( ... ) {
    if ( 条件式 ) {
        break ;
    }
}
```



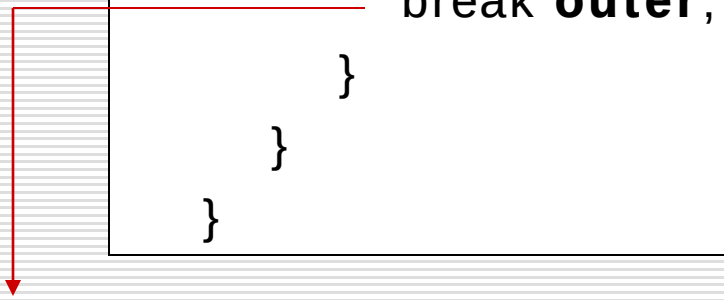
ループ終了

～ C言語と同じ

# while文とString型（第7章）

- ラベル付きbreak文
  - 多重ループの中断

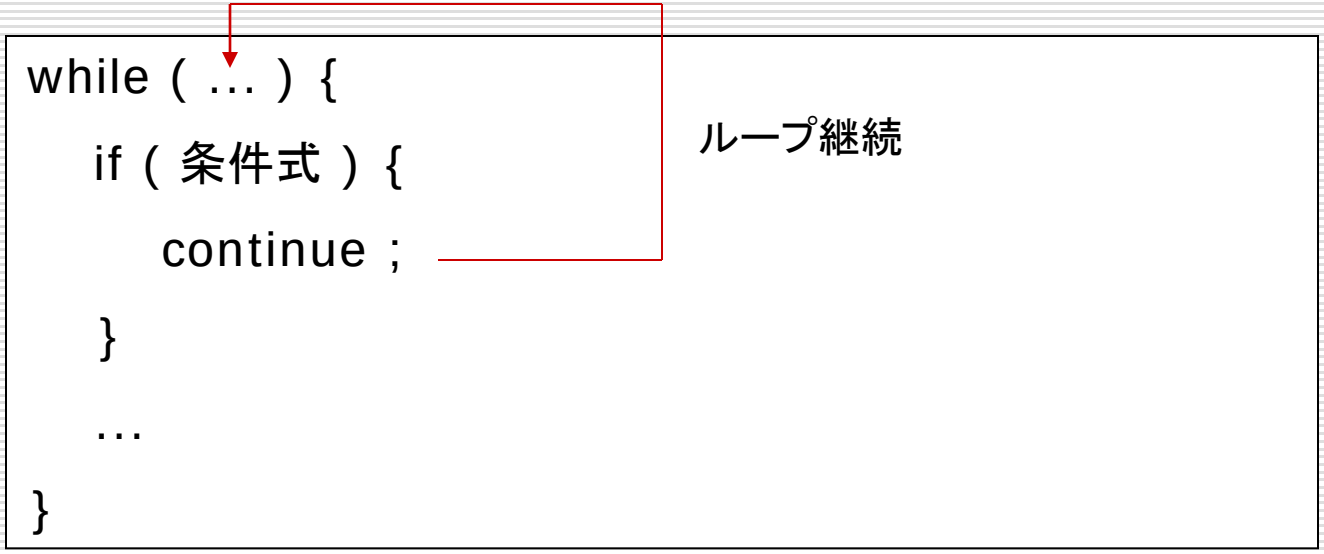
```
outer:
    while ( ... ) {
        while ( ... ) {
            if ( 条件式 ) {
                break outer;
            }
        }
    }
```



ループ終了

# while文とString型（第7章）

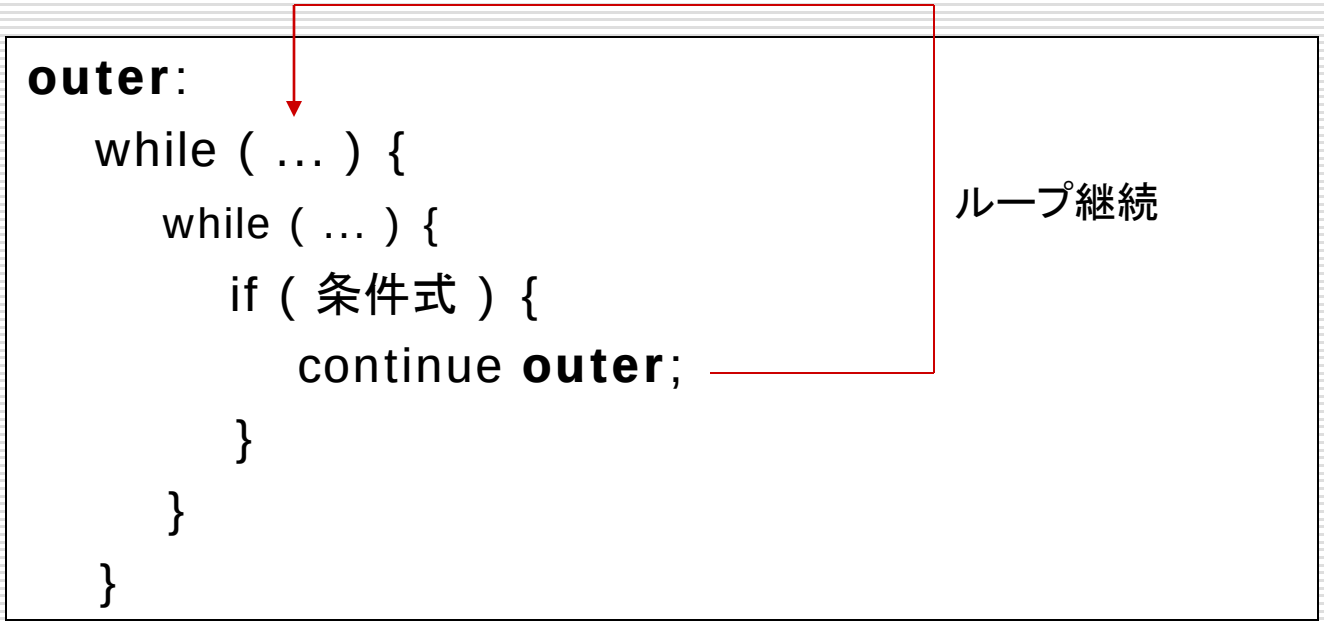
- continue文
  - 繰り返しの継続




～ C言語と同じ

# while文とString型（第7章）

- ラベル付きcontinue文
  - 多重ループの継続



# while文とString型（第7章）

- if 文と while 文

```
if ( 条件式 ) {
        実行文 ;
}
```

```
while ( 条件式 ) {
        実行文 ;
}
```

  - if 文は、繰り返さない while 文