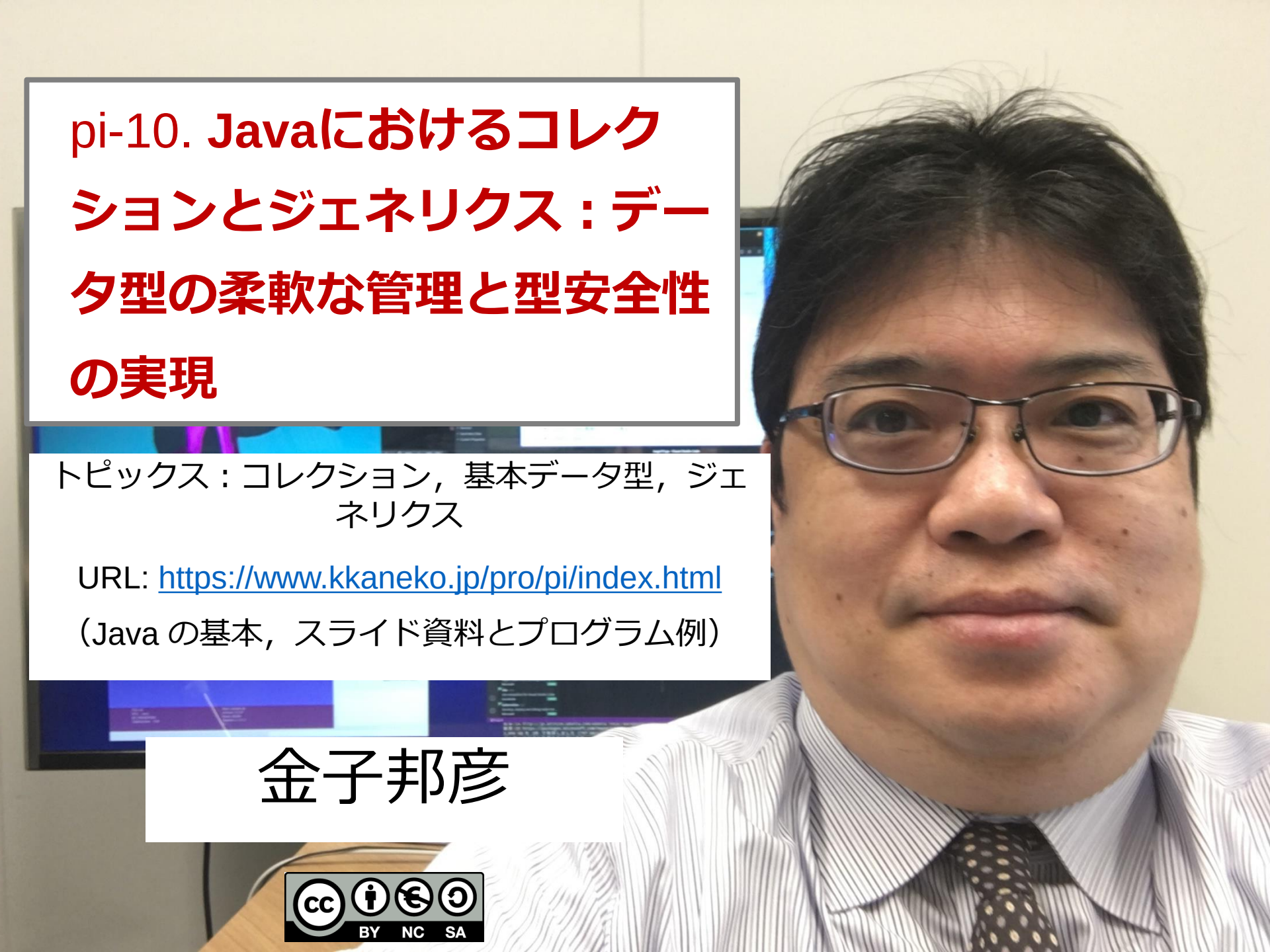
pi-10. Javaにおけるコレクションとジェネリクス：データ型の柔軟な管理と型安全性の実現

トピックス：コレクション，基本データ型，ジェネリクス

URL: https://www.kkaneko.jp/pro/pi/index.html

（Java の基本，スライド資料とプログラム例）



金子邦彦

- **オブジェクトの集まりを１つのオブジェクトとして扱いたい**

→　コレクション

Java の標準機能（ArrayList, HashMap など）

実際に使うときの注意点を説明

全体まとめ

- **Java の基本データ型**のそれぞれに，**ラッパクラス**がある．
- Java では「**基本データ型**の**コレクション**」というものは**ない**

  ArrayList<int> はない

  「**ラッパクラス**の**コレクション**」で**代用**

  ArrayList<Integer> で代用
- **Java の標準機能**として，**ArrayList**，**HashMap** など，**コレクション**を扱うための**クラス**がある．
- Java の**コレクションのクラス**の基礎は，**ジェネリクス**である

アウトライン

番号	項目
	復習
10-1	Java のコレクション
10-2	Java の基本データ型
10-3	ジェネリクス

各自、資料を読み返したり、課題に取り組んだりも行う

この授業では、**Java** を用いて基礎を学び、マスターする

Java Tutor の起動

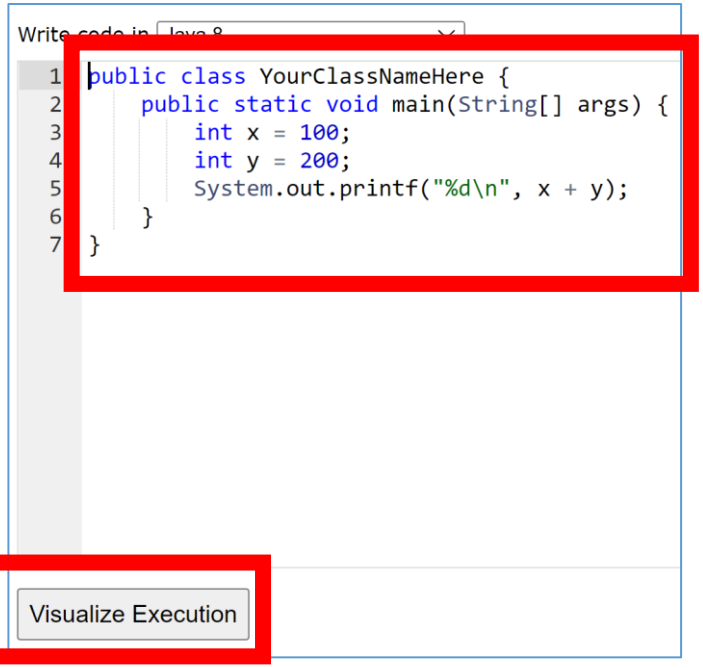
① **ウェブブラウザ**を起動する

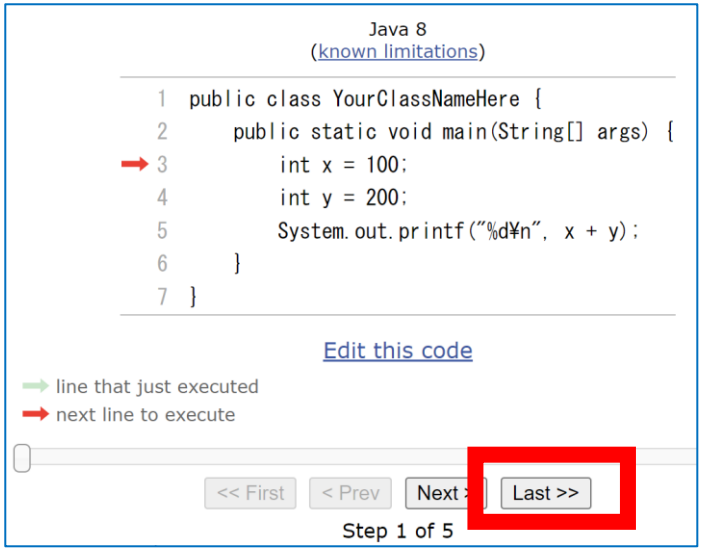
② **Java Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

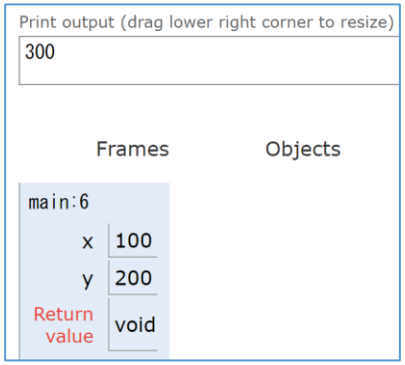
③ 「**Java**」をクリック　⇒ **編集画面**が開く

Java Tutor でのプログラム実行手順



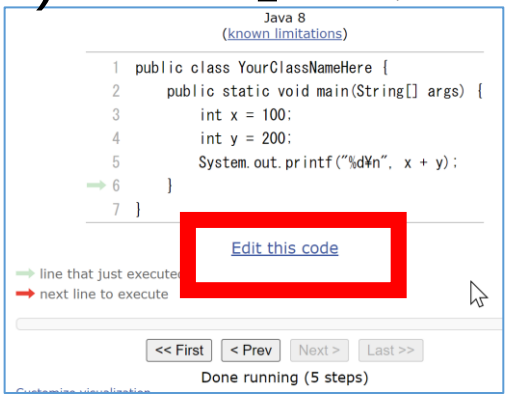


(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える



(3) **<u>実行結果を確認</u>**する.

(2)「**Last**」をクリックする.



(4)「**Edit this code**」をクリックして**編集画面**に戻る

Java Tutor 使用上の注意点①

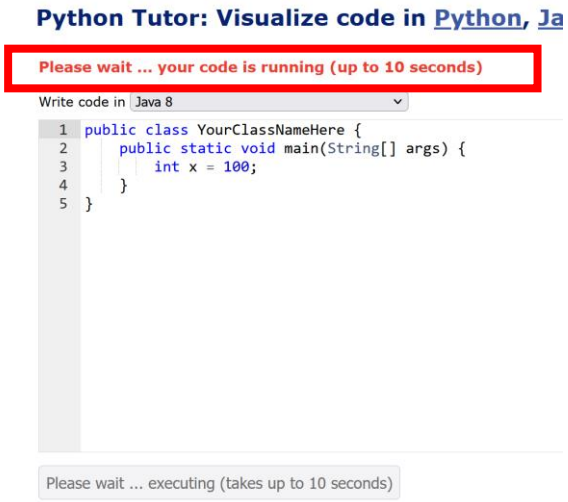
- **実行画面で，次のような赤の表示が出ることがある → 無視してよい**

  過去の文法ミスに関する確認表示

  邪魔なときは「**Close**」

Java Tutor 使用上の注意点②

「**please wait ... executing**」のとき，**１０秒ほど待つ**.



→ **混雑しているとき**は，**「Server Busy・・・」というメッセージが出る**ことがある.

混雑している．**少し（数秒から数十秒）待つ**と自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

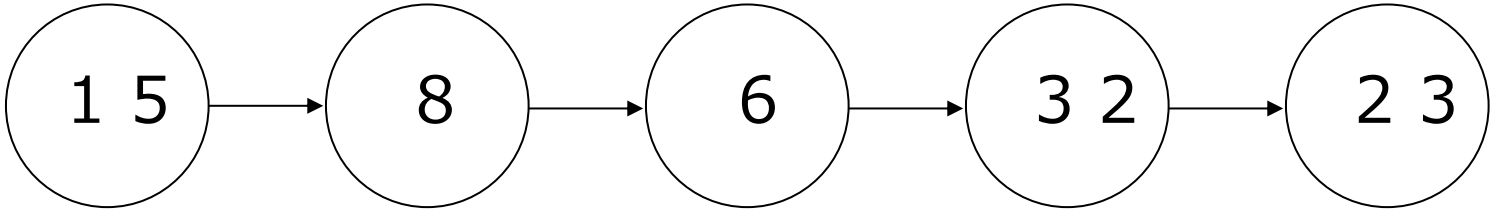
10-1. Java のコレクション

コレクションとは

- **コレクション**は、**オブジェクトの集まりを１つのオブジェクトとして扱いたい**ときに使う

- **Java の標準機能**として，**ArrayList**，**HashMap**など，**コレクション**を扱うための**クラス**がある

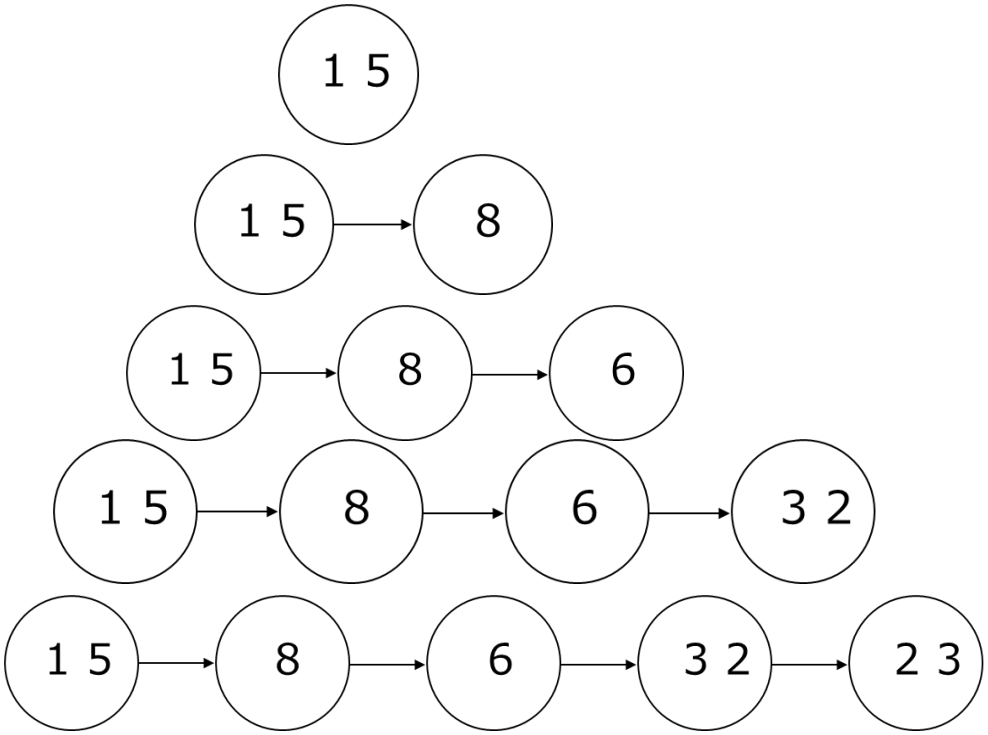
リスト (List) とは



- **順序のあるデータ**
- 要素の**削除**，**挿入**により**サイズが増減**する

add メソッドは、要素の挿入

```
m.add("15");
m.add("8");
m.add("6");
m.add("32");
m.add("23");
```



演習

資料：**14 ～ 15**

【トピックス】
・リスト

① Java Tutor のエディタで次のプログラムを入れる


```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4  public class YourClassNameHere {
5      public static void main(String[] args) {
6          ArrayList<String> m = new ArrayList<String>();
7          m.add("15");
8          m.add("8");
9          m.add("6");
10         m.add("32");
11         m.add("23");
12         for(String s : m) {
13             System.out.println(s);
14         }
15     }
16 }
```

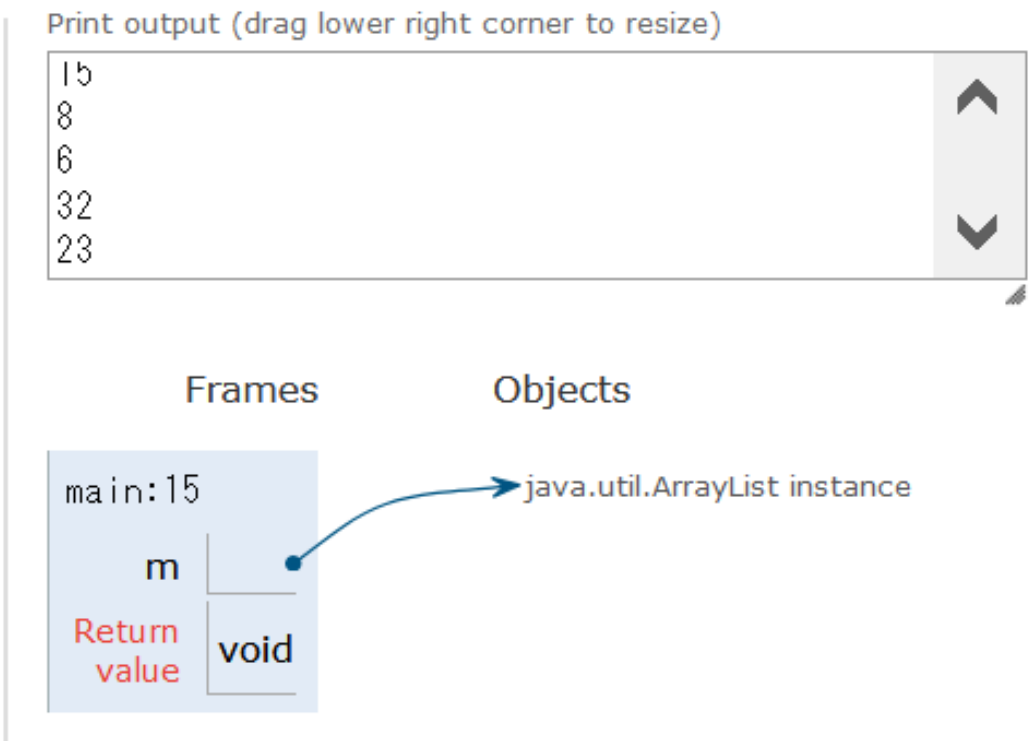
空のリストの
組み立て

add メソッドは
要素の挿入

拡張 for 文で
リストの要素をたどる

② 実行し，結果を確認する

（あとで使うので、プログラムを消さないこと）



15, 8, 6, 32, 23 が表示される

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

Java でのコレクションの使い方

- **型変数**を使う（**コレクション**に入れる**オブジェクト**の**クラス**を示す）

```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4  public class YourClassNameHere {
5      public static void main(String[] args) {
6        ArrayList<String> m = new ArrayList<String>();
7        m.add("15");
8        m.add("8");
9        m.add("6");
10       m.add("32");
11       m.add("23");
12       for(String s : m) {
13         System.out.println(s);
14       }
15     }
16 }
```

- **異なるクラス**の**オブジェクト**を**コレクション**で扱いたいときは、**共通のスーパークラス**を、**型変数**に指定

演習

資料：**18 ～ 19**

【トピックス】

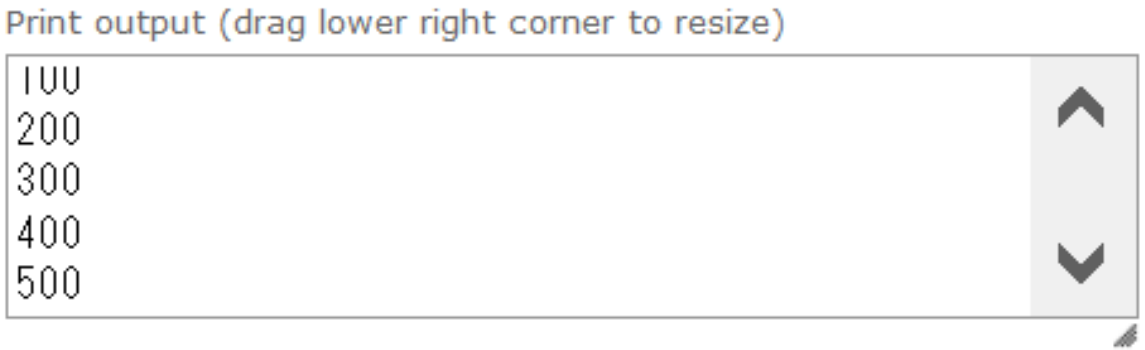
- **リスト**
- **型変数**

① Java Tutor のエディタで次のプログラムを入れる

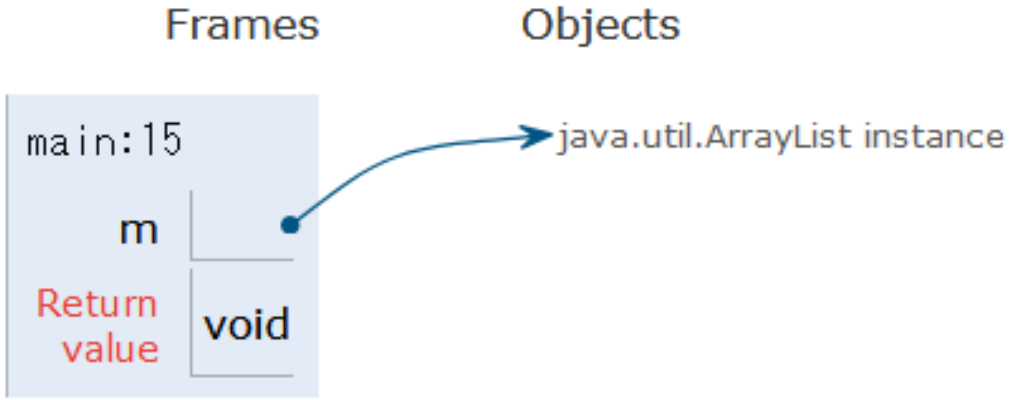
```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4  public class YourClassNameHere {
5      public static void main(String[] args) {
6        ArrayList<Integer> m = new ArrayList<Integer>();
7        m.add(100);
8        m.add(200);
9        m.add(300);
10       m.add(400);
11       m.add(500);
12       for(Integer s : m) {
13         System.out.println(s);
14       }
15     }
16 }
```

② 実行し，結果を確認する

（あとで使うので、プログラムを消さないこと）



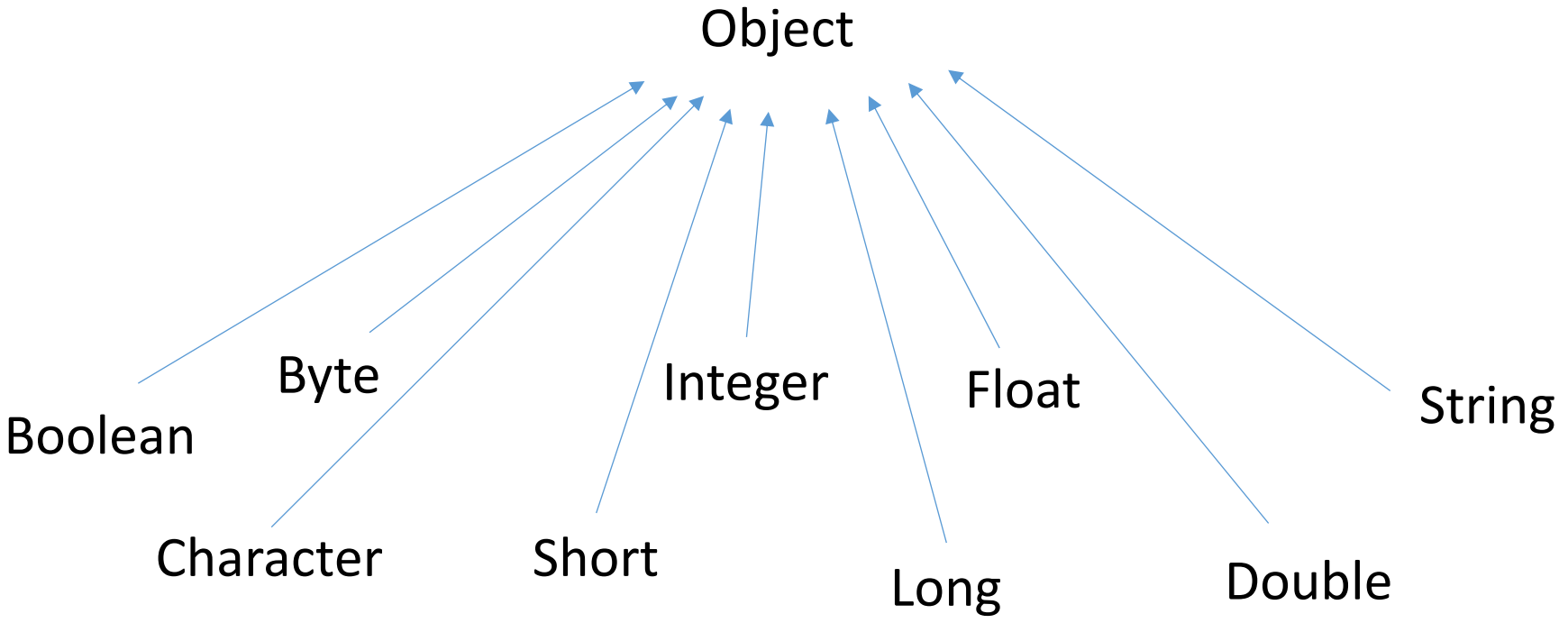
100, 200, 300, 400, 500 が表示される



「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

クラス階層のイメージ

Java の標準機能の**クラス**の多くは、Object の**サブクラス**



String, Integer に**共通するスーパークラス**は Object

演習

資料：**22 ～ 23**

【トピックス】

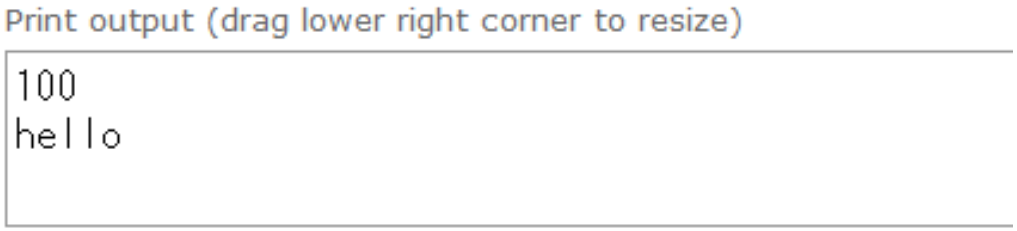
- **リスト**
- **型変数**
- **クラス階層**

① Java Tutor のエディタで次のプログラムを入れる

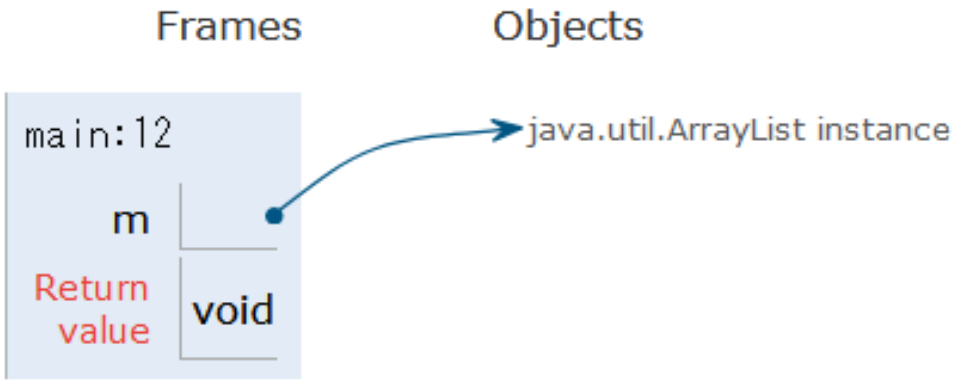

```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4  public class YourClassNameHere {
5      public static void main(String[] args) {
6          ArrayList<Object> m = new ArrayList<Object>();
7          m.add(100);
8          m.add("hello");
9          for(Object s : m) {
10             System.out.println(s);
11         }
12     }
13 }
```

② 実行し，結果を確認する

（あとで使うので、プログラムを消さないこと）



100, hello が表示される



「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

型変数は、毎回「**Object**」と書けば大丈夫ですか？

```
ArrayList<Object> m = new ArrayList<Object>();
```

いいえ。
・プログラムが分かりにくくなる
・不便になる

型変数には、なるべく**具体的なクラス名を書いた方がよい**です.

10-2. Java の基本データ型

Java のデータの種類

- **基本データ**

データの種類	**基本データ型**	サイズ
整数	**byte**	8 bit
	short	16 bit
	int	32 bit
	long	64 bit
浮動小数	**float**	32 bit
	double	64 bit
文字	**char**	16 bit
true/false	**boolean**	

- **基本データ**の**配列**
- **クラス**に属する**オブジェクト**:
  Javaの標準機能のクラスは,
  String, ArrayList, HashMap クラスなど多種

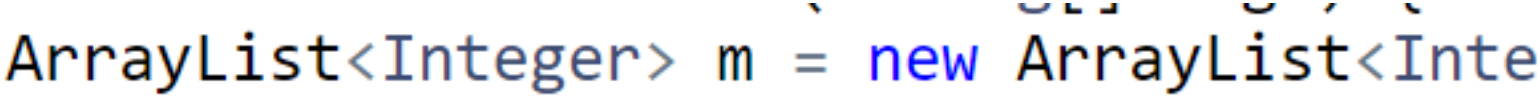
Java のコレクションは、オブジェクトのコレクション

- Java では「**基本データ型**の**コレクション**」というものは**ない**

  ArrayList<int> などはない

データの種類	**基本データ型**	サイズ
整数	**byte**	8 bit
	short	16 bit
	int	32 bit
	long	64 bit
浮動小数	**float**	32 bit
	double	64 bit
文字	**char**	16 bit
true/false	**boolean**	

Java のコレクションは、オブジェクトのコレクション



これは OK

これは**動かない**

Java のラッパクラス

基本データ型に**対応した**クラス

ラッパクラス	基本データ型
Boolean	boolean
Character	char
Byte	byte
Short	short
Integer	int
Long	long
Float	float
Double	double

Java のコレクションは、オブジェクトのコレクション

- Java では「**基本データ型**の**コレクション**」というものは**ない**

  ArrayList<int> はない

「**ラッパクラス**の**コレクション**」で**代用**

  ArrayList<Integer> で代用

10-3. Java のジェネリクス

Java のジェネリクスとは

- **クラス名の後**に「**<クラス名>**」を**指定**して、**種々の制限**などができる機能

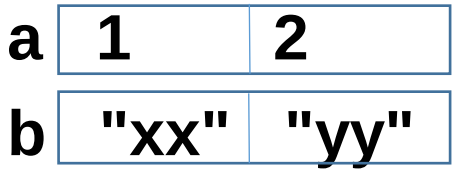
```
ArrayList<Integer> m = new ArrayList<Integer>();
```

ArrayList は，コレクションのクラス.

ArrayList<Integer> は，m には**整数しか入れない**という制限

Java のオブジェクトの生成

次の 2 つの**オブジェクトを生成する** Java プログラム




```
Pair a = new Pair<Integer>(1, 2);
Pair b = new Pair<String>("xx", "yy");
```

さまざまなクラスのオブジェクトが入る.

s と e のクラスは同じに**制限**したい.

- このとき，次の**クラス**を使うことにする

クラス名 **Pair**

属性 s, e

```
1  class Pair<T> {
2    T s;
3    T e;
4    public Pair(T s, T e) {
5      this.s = s;
6      this.e = e;
7    }
8  }
```

```
1  class Pair<T> {
2    T s;
3    T e;
4    public Pair(T s, T e) {
5      this.s = s;
6      this.e = e;
7    }
8  }
```

- クラス定義で「**class Pair<T>**」のような書き方ができる.
- このとき、コンストラクタで「**new Pair<Integer>**」のような書き方ができる.

演習

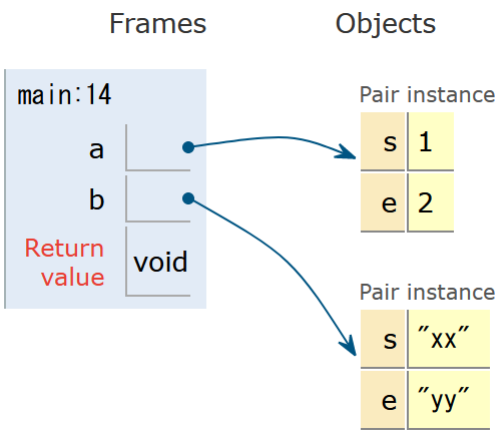
資料：**36 ～ 37**

【トピックス】

・ジェネリクス

① Java Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

```
1  class Pair<T> {
2    T s;
3    T e;
4    public Pair(T s, T e) {
5      this.s = s;
6      this.e = e;
7    }
8  }
9
10 public class YourClassNameHere {
11     public static void main(String[] args) {
12       Pair a = new Pair<Integer>(1, 2);
13       Pair b = new Pair<String>("xx", "yy");
14     }
15 }
```



表示を確認

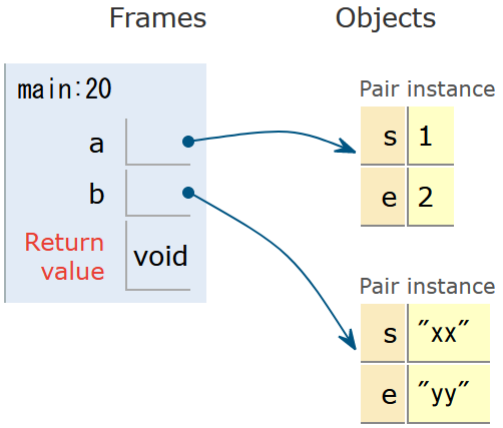
「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

② Java Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

```
1  class Pair<T> {
2    T s;
3    T e;
4    public Pair(T s, T e) {
5      this.s = s;
6      this.e = e;
7    }
8    public void print() {
9      System.out.println(s);
10     System.out.println(e);
11   }
12 }
13
14 public class YourClassNameHere {
15     public static void main(String[] args) {
16       Pair a = new Pair<Integer>(1, 2);
17       Pair b = new Pair<String>("xx", "yy");
18       a.print();
19       b.print();
20     }
21 }
```

Print output (drag lower right corner to resize)

```
1
2
xx
yy
```



表示を確認

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

全体まとめ

- **Java の基本データ型**のそれぞれに，**ラッパクラス**がある．
- Java では「**基本データ型**の**コレクション**」というものは**ない**

  ArrayList<int> はない

  「**ラッパクラス**の**コレクション**」で**代用**

  ArrayList<Integer> で代用
- **Java の標準機能**として，**ArrayList**，**HashMap** など，**コレクション**を扱うための**クラス**がある．
- Java の**コレクションのクラス**の基礎は，**ジェネリクス**である

関連ページ

- **Java プログラミング入門**

GDB online を使用

https://www.kkaneko.jp/pro/ji/index.html

- **Java の基本**

Java Tutor, GDB online を使用

https://www.kkaneko.jp/pro/pi/index.html

- **Java プログラム例**

https://www.kkaneko.jp/pro/java/index.html

ソースコード 10-1


```
import java.util.ArrayList;
import java.util.Iterator;

public class YourClassNameHere {
    public static void main(String[] args) {
      ArrayList<String> m = new ArrayList<String>();
      m.add("15");
      m.add("8");
      m.add("6");
      m.add("32");
      m.add("23");
      for(String s : m) {
        System.out.println(s);
      }
    }
}
```

ソースコード 10-1


```
import java.util.ArrayList;
import java.util.Iterator;

public class YourClassNameHere {
    public static void main(String[] args) {
      ArrayList<Integer> m = new ArrayList<Integer>();
       m.add(100);
       m.add(200);
       m.add(300);
       m.add(400);
       m.add(500);
       for(Integer s : m) {
         System.out.println(s);
       }
    }
}
```

ソースコード 10-3


```
class Pair<T> {
  T s;
  T e;
  public Pair(T s, T e) {
    this.s = s;
    this.e = e;
  }
  public void print() {
    System.out.println(s);
    System.out.println(e);
  }
}

public class YourClassNameHere {
    public static void main(String[] args) {
      Pair a = new Pair<Integer>(1, 2);
      Pair b = new Pair<String>("xx", "yy");
      a.print();
      b.print();
    }
}
```