pi-5. Javaにおけるコレクション：リストとマップ

（Java マスター講座：基礎から応用まで）

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

金子邦彦

CC BY NC SA

全体まとめ

- **複数データを扱うときの方法**

単一の**コレクション**（リストやマップなど）で扱うことができる.

- **Java の標準機能**に，**コレクション**がある.
- 検索，挿入，削除などを行いたい場合，単一の**コレクション**（オブジェクト）で扱う方が楽である.
- **リスト**は，**順序のあるデータ**である．**要素の削除，挿入によりサイズが増減**する
- **マップ**は，**キー**と**値（バリュー）**の**ペア**の集まりである．**同じ値のキーは２回以上登場しない**

アウトライン

番号	項目
	復習
5-1	リスト
5-2	リストを演習できるオンラインサイトの紹介
5-3	マップ

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

オブジェクトとメソッド

hero.moveDown()

hero　　オブジェクト
moveDown()　メソッド
間を「.」で**区切って**いる

- **メソッド: オブジェクト**に属する操作や処理.
- **メソッド**呼び出しでは，**引数**を指定することがある．**引数**（ひきすう）は，**メソッド**に渡す値のこと

hero.attack("fence", 36, 26)

Java プログラムの書き方

プログラムの例

```
x = 100
a = x + 200
enermy1 = hero.findNearestEnemy()
hero.attack(enemy1)
```

- **代入**：**オブジェクト名** ＋ 「**=**」
  ＋ 式または値またはメソッド呼び出し
- **メソッドアクセス**：**オブジェクト名** ＋ 「**.**」
  ＋ **メソッド名** ＋「**()**」 （引数を付けることも）

その他，属性アクセス，関数呼び出し，制御，「*」，「+」などの演算子，コマンド，定義など

Java Tutor の起動

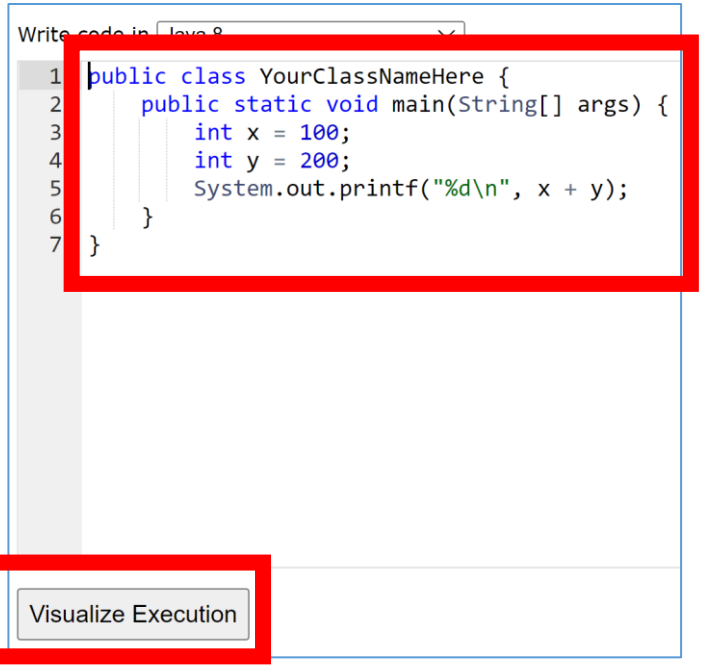
① **ウェブブラウザ**を起動する

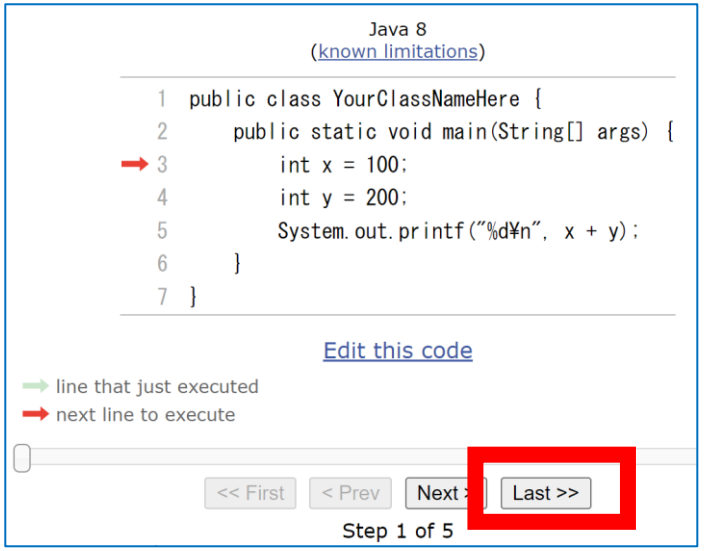
② **Java Tutor** を使いたいので，次の URL を開く

https://pythontutor.com/java.html

③ 「**Java**」をクリック　⇒ **編集画面**が開く

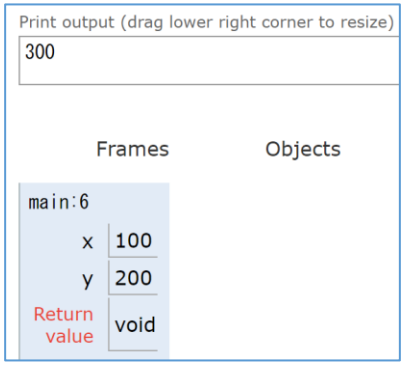
Java Tutor でのプログラム実行手順

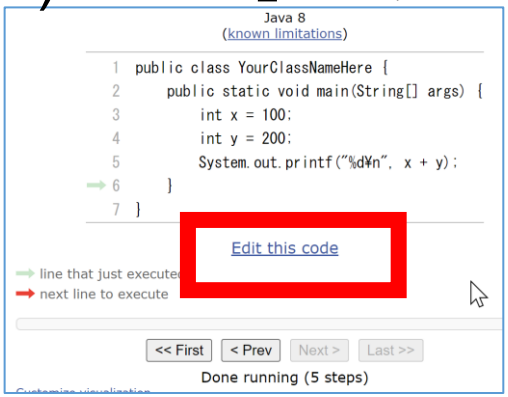




(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える

(2)「**Last**」をクリック.





(3) **<u>実行結果を確認</u>**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

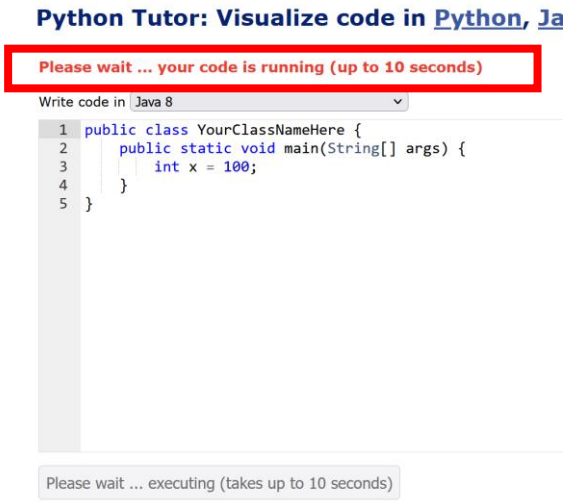
- **実行画面で，次のような赤の表示が出ることがある → 無視してよい**

  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**.



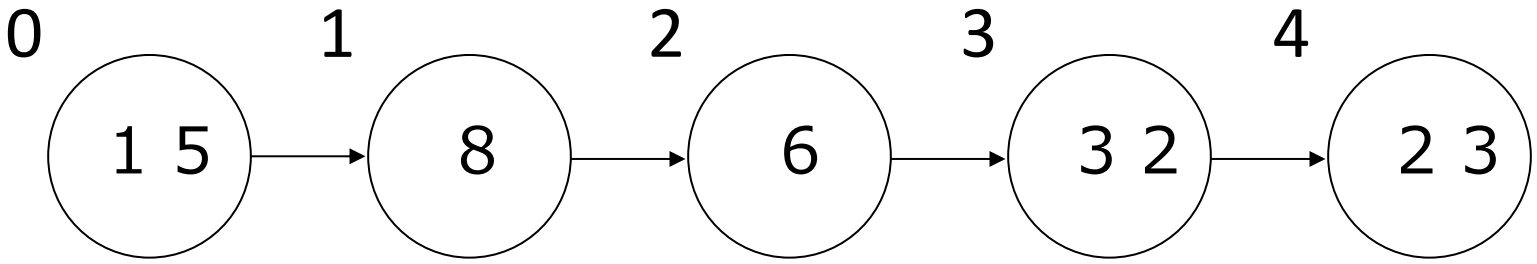
→ **混雑しているとき**は，**「Server Busy・・・」というメッセージが出る**ことがある.

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）

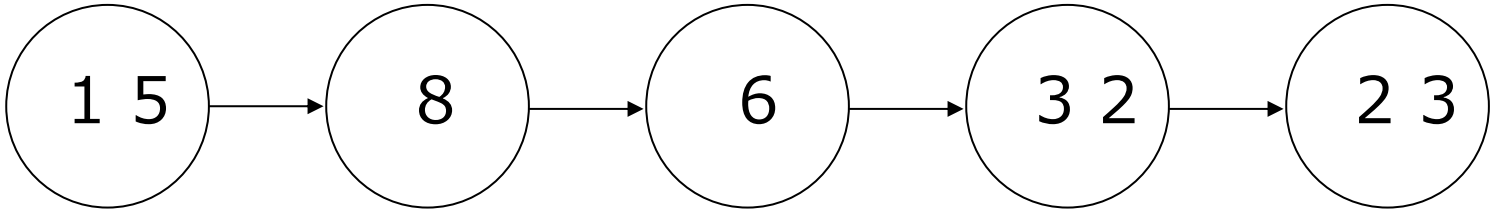
5-1. リスト

リスト

- **リスト**は，**同じ型の要素の並び**
- **リスト**の要素には**順序がある**．０から始まる番号（**添字**）が付いている
- 要素の**削除**，**挿入**により**サイズが増減**する



リストを生成するプログラム



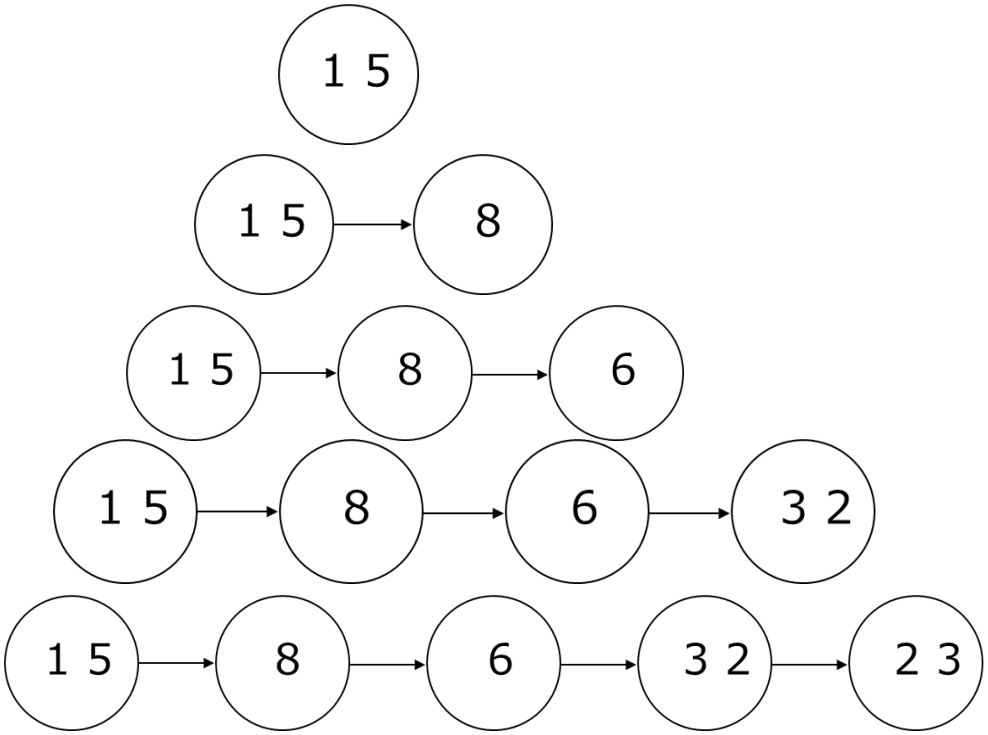

```
1  import java.util.ArrayList;
2
3  public class YourClassNameHere {
4      public static void main(String[] args) {
5          ArrayList<String> m = new ArrayList<String>();
6          m.add("15");
7          m.add("8");
8          m.add("6");
9          m.add("32");
10         m.add("23");
11 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

空のリストの組み立て

add メソッドは要素の挿入

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：**15 ～ 21**

【トピックス】

- **リスト**
- **ArrayList**

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
3  public class YourClassNameHere {
4      public static void main(String[] args) {
5          ArrayList<String> m = new ArrayList<String>();
6          m.add("15");
7          m.add("8");
8          m.add("6");
9          m.add("32");
10         m.add("23");
11     }
12 }
```

空のリストの組み立て

add メソッドは要素の挿入

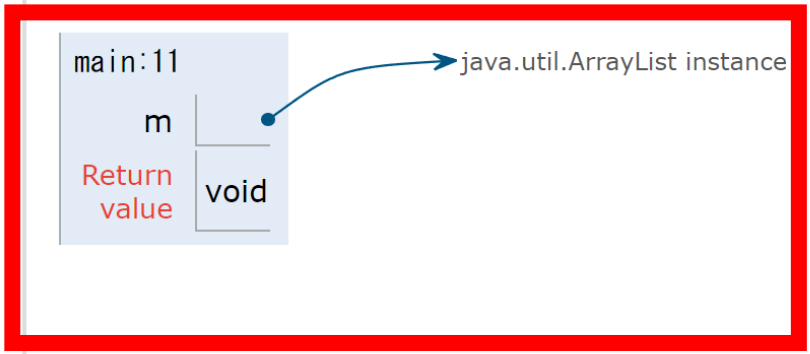
② 実行し，結果を確認する

（プログラムはあとで使うので消さないこと）

Java 8
(known limitations)

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

Edit this code



オブジェクト m が生成される．
（JavaTutor には，コレクションの要素を表示する機能はない）

「**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 の ArrayList クラス

- 標準機能として，次の**メソッド**がある.

コンストラクタ	ArrayList
要素の挿入	add
要素の削除	clear, remove
要素の取得	get
検索	indexOf
要素値の更新	set
要素数	size

など

リストの機能

③ Java Tutor のエディタで次のプログラムを入れる

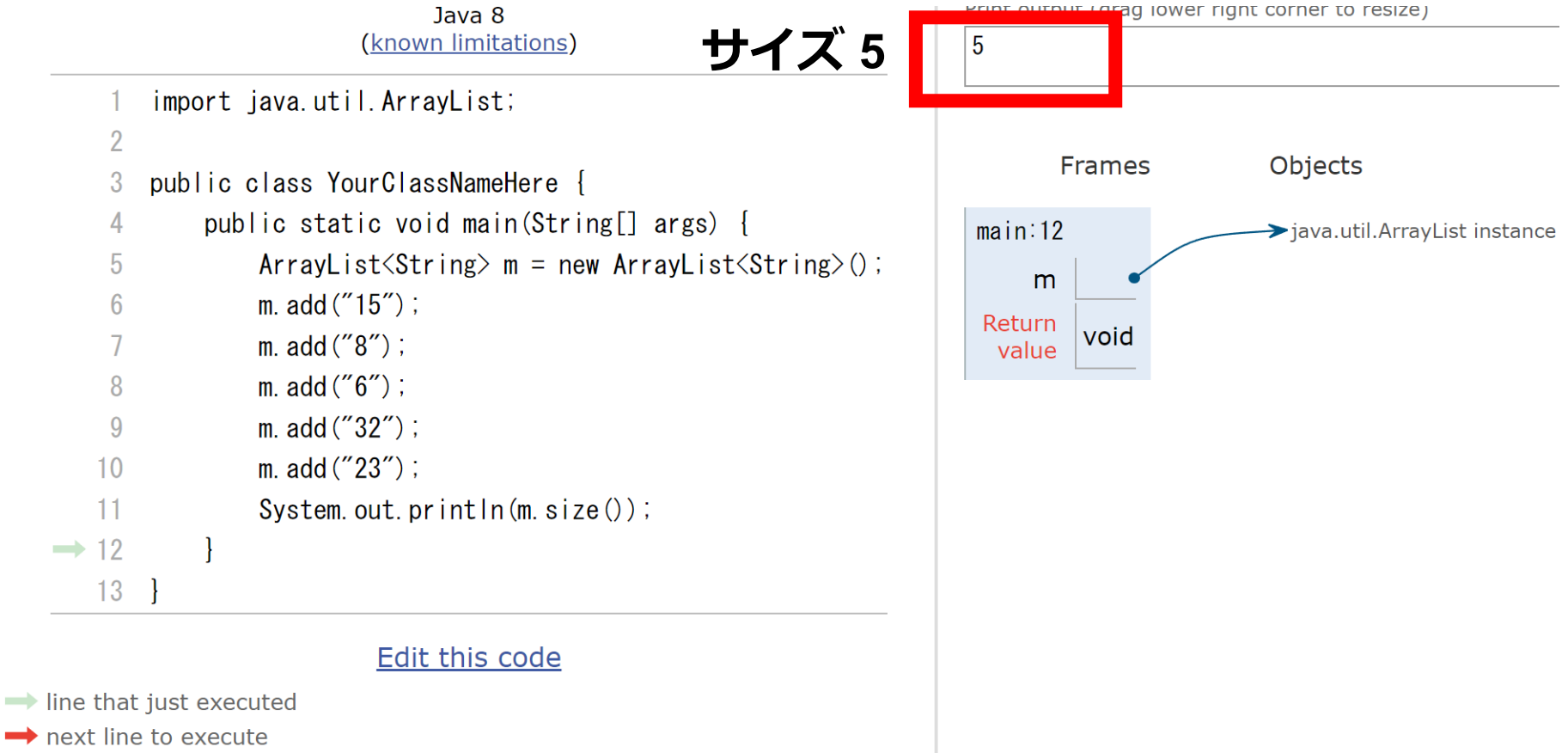
size で要素数を数える

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

この中で
メソッド size の呼び出し

④ 実行し，結果を確認する

(プログラムはあとで使うので消さないこと)



「**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

⑤ Java Tutor のエディタで次のプログラムを入れる

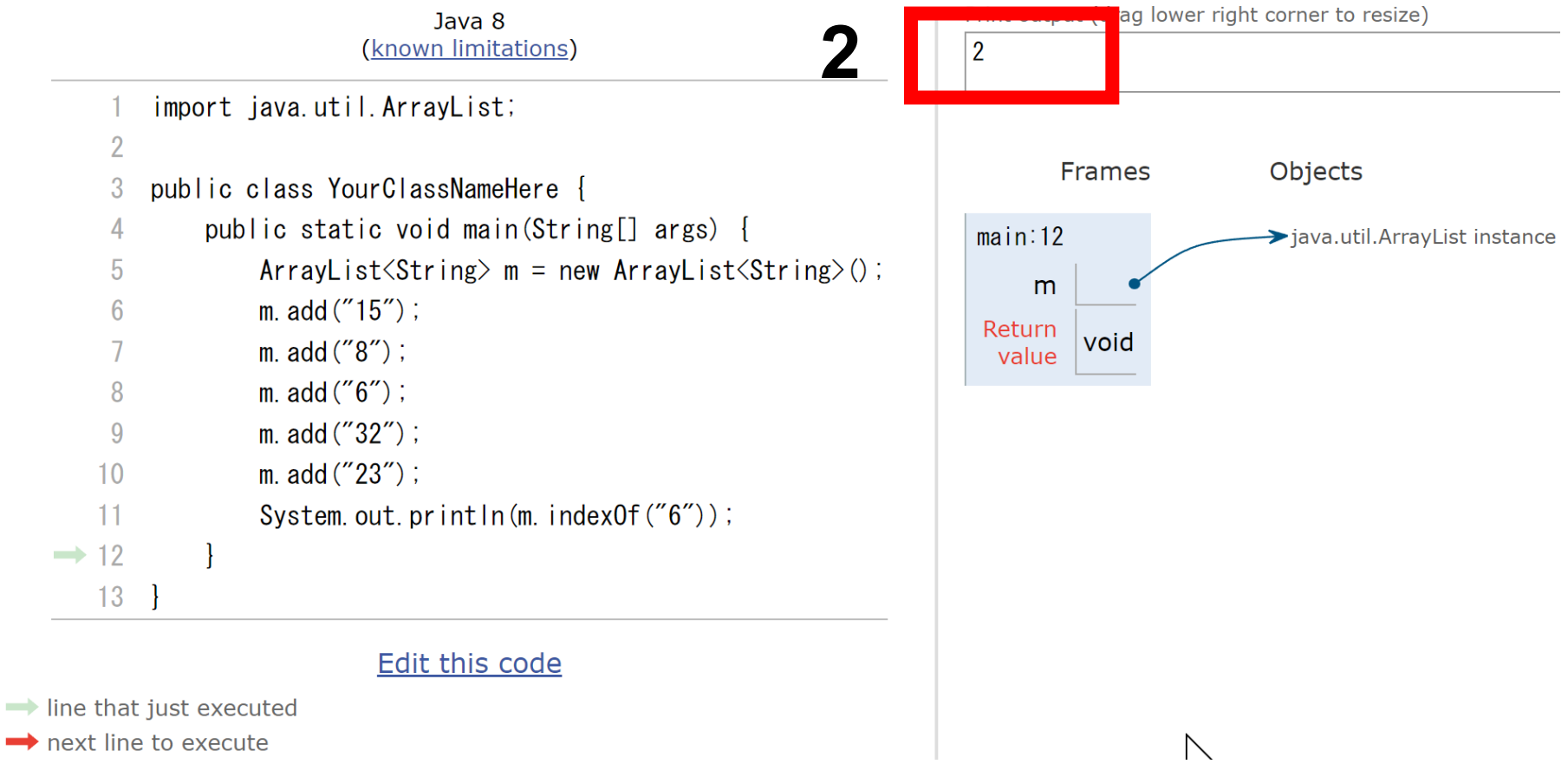
indexOf で，リストの中から「"6"」を探す．

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

この中で
メソッド indexOf の呼び出し

⑥ 実行し，結果を確認する

(プログラムはあとで使うので消さないこと)



「**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

5-2. リストを演習できるオンラインサイトの紹介

「リスト」を演習できる オンラインサイトの紹介

リストは，同じ型の要素の並び

① ウェブブラウザを起動する

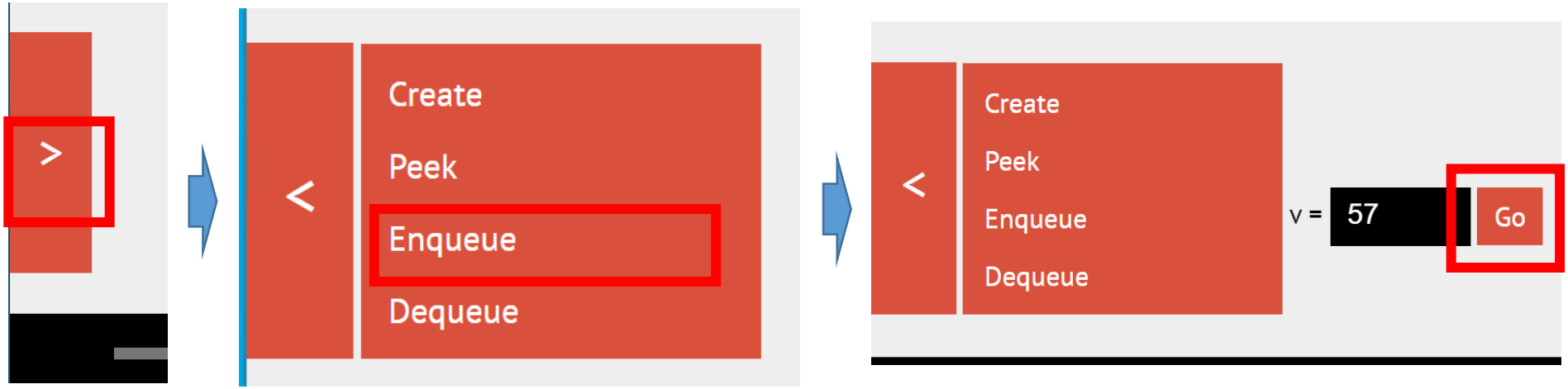
② 次の URL を開く

https://visualgo.net/ja

③ 「**Linked List** (連結リスト)」をクリック

④ 説明が出る場合がある．**ESC キー**を押して，説明を消す

⑤　左下のメニューで「**Enqueue** (入れる)」をクリックし，「**Go**」をクリック



⑥　**<u>末尾に挿入される</u>**ので，確認する

5-3. マップ

マップ (Map)

キー	値（バリュー）
1	Red
2	Yellow
3	Blue

- **マップ**は，**キー**と**値（バリュー）**の**ペア**の集まり
- 同じ値の**キー**は**2回以上登場しない**

マップを生成するプログラム

```
1  import java.util.HashMap;
2
3  public class YourClassNameHere {
4      public static void main(String[] args) {
5          HashMap<Integer, String> x = new HashMap<Integer, String>();
6          x.put(1, "Red");
7          x.put(2, "Yellow");
8          x.put(3, "Blue");
9          System.out.println(x.size());
10     }
11 }
```

空のマップの
組み立て

「put」は
要素の挿入

この中で
メソッド size の呼び出し

演習

資料：**31 ～ 36**

【トピックス】

- **マップ**
- **HashMap**

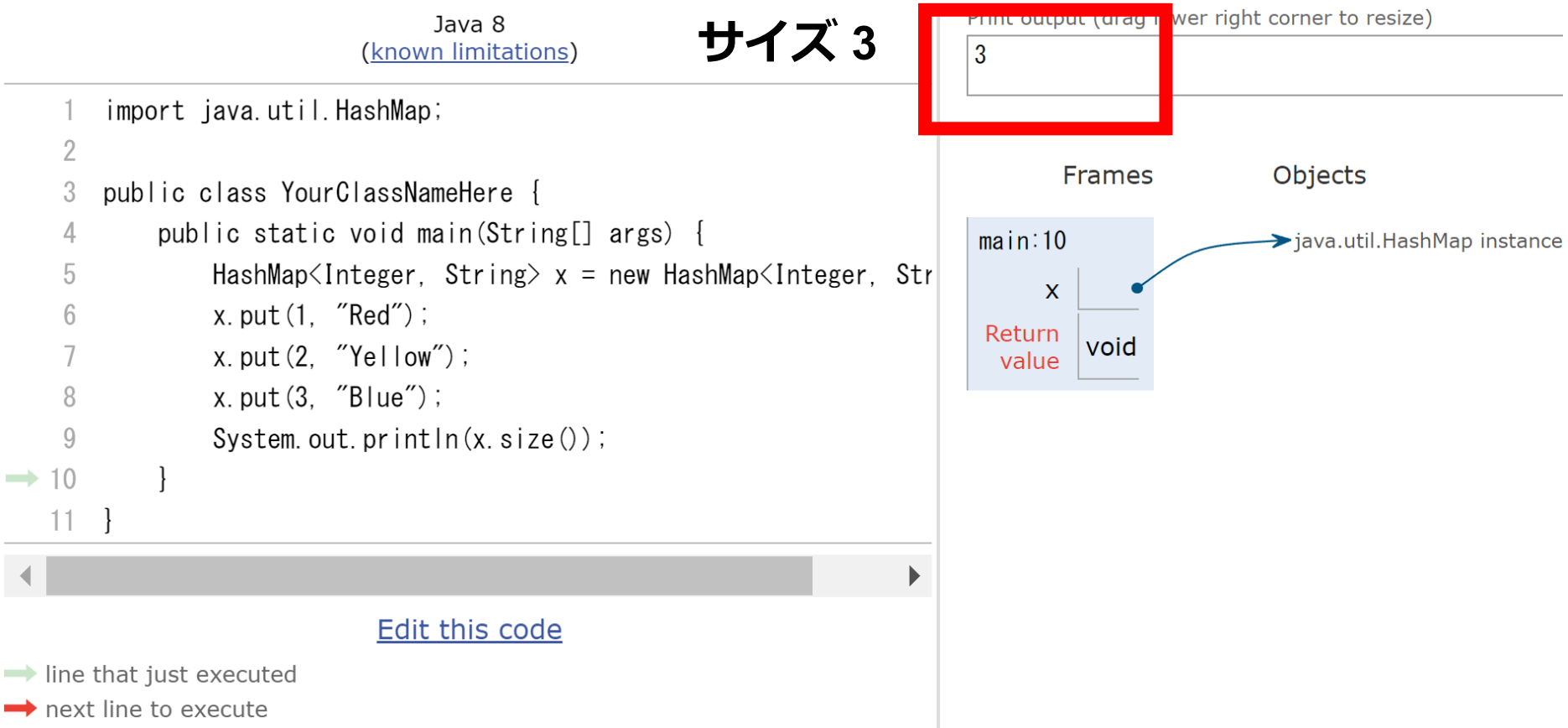
① Java Tutor のエディタで次のプログラムを入れる

```java
import java.util.HashMap;

public class YourClassNameHere {
    public static void main(String[] args) {
        HashMap<Integer, String> x = new HashMap<Integer, String>();
        x.put(1, "Red");
        x.put(2, "Yellow");
        x.put(3, "Blue");
        System.out.println(x.size());
    }
}
```

② 実行し，結果を確認する

(プログラムはあとで使うので消さないこと)



「**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 の HashMap クラス

- 標準機能として，次のメソッドがある．便利.

コンストラクタ	HashMap
要素の挿入	put
要素の削除	clear, remove
検索	get
要素数	size
など	

マップの機能．ペアを扱うときに便利.

Java の HashMap クラスの get メソッド

キー	値（バリュー）
1	Red
2	Yellow
3	Blue

- **get メソッド**

キーから**値（バリュー）**を得る

x.get(**1**)　→　結果は「**Red**」
x.get(**2**)　→　結果は「**Yellow**」
x.get(**3**)　→　結果は「**Blue**」

③ Java Tutor のエディタで次のプログラムを入れる

```java
import java.util.HashMap;

public class YourClassNameHere {
    public static void main(String[] args) {
        HashMap<Integer, String> x = new HashMap<Integer, String>();
        x.put(1, "Red");
        x.put(2, "Yellow");
        x.put(3, "Blue");
        System.out.println(x.get(2));
    }
}
```

④ 実行し，結果を確認する

(プログラムはあとで使うので消さないこと)

「**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

マップの生成，要素の挿入，拡張 For文

まとめページ

- **マップの生成**

```
HashMap<Integer, String> x = new HashMap<Integer, String>();
```

x: オブジェクト名

Integer:
キーの**クラス**名

String:
値（バリュー）の
クラス名

- **要素の挿入**

```
x.put(1, "Red");
```

x : オブジェクト名　1, “Red”: 挿入したい要素

全体まとめ

- **複数データを扱うときの方法**

単一の**コレクション**（リストやマップなど）で扱うことができる.

- **Java の標準機能**に，**コレクション**がある.
- 検索，挿入，削除などを行いたい場合，単一の**コレクション**（オブジェクト）で扱う方が楽である.
- **リスト**は，**順序のあるデータ**である．**要素の削除，挿入によりサイズが増減**する
- **マップ**は，**キー**と**値（バリュー）**の**ペア**の集まりである．**同じ値のキーは２回以上登場しない**

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

資料中のソースコード 5-1

```
import java.util.ArrayList;

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");
        System.out.println(m.size());
    }
}
```

資料中のソースコード 5-3

```
import java.util.HashMap;

public class YourClassNameHere {
    public static void main(String[] args) {
        HashMap<Integer, String> x = new HashMap<Integer, String>();
        x.put(1, "Red");
        x.put(2, "Yellow");
        x.put(3, "Blue");
        System.out.println(x.size());
    }
}
```