Java Programming

2-4: Collections – Part 1

Practice Activities

Lesson Objectives:

- Create a collection without using generics
- Create a collection using generics
- Implement an ArrayList
- Implement a Set

Vocabulary:

Identify the vocabulary word for each definition below.

HashSet	A set similar to an ArrayList without any specific ordering.
List	An ordered Collection that may contain duplicates.
Collection	An interface used to define a group of objects. This includes lists and sets.
ArrayList	A list that is very similar to an array.
Set	A Collection of elements that does not contain any duplicates.

Try It/Solve It:

1. What is the difference between a set and a list?

   Perbedaan utamanya adalah:

   - **List**: Sebuah List adalah koleksi *terurut* (ordered) dan *dapat berisi* elemen duplikat.
   - **Set**: Sebuah Set adalah koleksi yang *tidak berisi* elemen duplikat. Implementasi umumnya, HashSet, *tidak memiliki* urutan tertentu.

2. You decide you want to roll 2 dice and see what the frequency is of each possible number combination. Would you use a Set collection to do this? State your reason(s).

   Tidak. Alasannya adalah Set tidak mengizinkan duplikat. Untuk menghitung frekuensi, perlu menyimpan *setiap* hasil lemparan dadu, termasuk hasil yang berulang (duplikat). Set hanya akan menyimpan satu kemunculan unik dari setiap kombinasi angka, sehingga mustahil untuk menghitung frekuensi. List, yang mengizinkan duplikat, akan lebih cocok.

3. Using a collection create a variable that will store a list of countries (Strings). Your collection should not store

duplicates, and order is not important. Test your code by adding 6 countries, one of which is a duplicate.

```java
import java.util.*;

public class CountrySTOR {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        HashSet<String> countries = new HashSet<String>();    Redundant typ

        countries.add(e:"Indonesia");
        countries.add(e:"Malaysia");
        countries.add(e:"Singapore");
        countries.add(e:"Thailand");
        countries.add(e:"Indonesia"); //duplicate, will not be added
        countries.add(e:"Vietnam");

        System.out.println(countries);

        System.out.println("Total countries: " + countries.size());
    }
}
```

The Output::

```
PS D:\SEMESTER 3 TEKNIK INFORMATIKA\PEMROGRAMAN-BERORIENT
A\PEMROGRAMAN-BERORIENTASI-OBJECT\GIT\Object-Oriented-SMT
OR }
[Vietnam, Singapore, Malaysia, Thailand, Indonesia]
Total countries: 5
PS D:\SEMESTER 3 TEKNIK INFORMATIKA\PEMROGRAMAN-BERORIENT
```

4. Would the following Collection.sort() statements both work? Explain your answer.

```java
HashSet<String> countriesSet = new HashSet<String>();
Collections.sort(countriesSet);
ArrayList<String> countriesList = new ArrayList();
Collections.sort(countriesList);
```

Tidak, kedua pernyataan tersebut tidak akan berfungsi:

1. Collections.sort(countriesSet);
    - **Tidak akan berfungsi.** Metode Collections.sort() memerlukan parameter berupa List. HashSet adalah implementasi dari Set, bukan List. Selain itu, HashSet pada dasarnya tidak memiliki urutan tertentu dan tidak dapat diurutkan.
2. Collections.sort(countriesList);

- **Akan berfungsi.** ArrayList adalah sebuah List. Metode Collections.sort() memang menerima List sebagai parameternya untuk penguruta

5. Below is a user implementation of a Stack using arrays.
   - push adds an item to the Stack
   - pop removes an item from the stack
   - isEmpty return a Boolean value of true if the Stack is empty

   Convert this to a generic implementation using an ArrayList.

```
public class ArrayStack {
	private int maxsize;
	private int top;
	private int[] items;

	public ArrayStack(int maxsize) {
		if (maxsize <= 0)
			throw new ArrayStackException(
					"Stack size must be positive");
		items = new int[maxsize];
		this.maxsize = maxsize;
		top = 0;
	}

	public void push(int item) {
		if (top == items.length)
			throw new ArrayStackException("Overflow Error");
		items[top] = item;
		top++;
	}


	public int pop() {
		if (isEmpty())
			throw new ArrayStackException("Underflow Error");
		return items[--top];
	}

	public boolean isEmpty() {
		return (top == 0);
	}

	public static class ArrayStackException extends RuntimeException {
		public ArrayStackException(String message) {
			super(message);
		}
	}


	public static void main(String[] args) {
		ArrayStack stack = new ArrayStack(3);
		stack.push(1);
		stack.push(2);
		stack.push(3);
		//stack.push(4);  //overflow error
		System.out.println(stack.pop());
		System.out.println(stack.pop());
		System.out.println(stack.pop());
```

```
        }
    }
```

Berikut adalah implementasi ulang ArrayStack menggunakan ArrayList generik. Tidak lagi memerlukan maxsize karena ArrayList dapat tumbuh dan menyusut secara dinamis.

```
import java.util.ArrayList;

public class GenericStack<T> { // Menjadi generik

    private ArrayList<T> itemsList;    Field itemsList can be final

    // Pengecualian tetap sama
    public static class ArrayStackException extends RuntimeException {
        public ArrayStackException(String message) {
            super(message);
        }
    }

    public GenericStack() {
        // Inisialisasi ArrayList
        itemsList = new ArrayList<T>();    Redundant type arguments in new
    }

    public void push(T item) {
        itemsList.add(item);
    }

    public T pop() {
        if (isEmpty()) {
            throw new ArrayStackException(message:"Underflow Error");
        }
        // Last-In, First-Out (LIFO)
        int lastIndex = itemsList.size() - 1;
        return itemsList.remove(lastIndex);
    }

    public boolean isEmpty() {
        // Stack kosong jika ukurannya 0
        return (itemsList.size() == 0);    itemsList.size() == 0 can be rep
    }
```

```
36
        Run | Debug | Run main | Debug main
37      public static void main(String[] args) {
38          // Menguji dengan Integer, seperti aslinya
39          GenericStack<Integer> stack = new GenericStack<Integer>();
40
41          stack.push(item:1);
42          stack.push(item:2);
43          stack.push(item:3);
44
45          System.out.println(stack.pop());
46          System.out.println(stack.pop());
47          System.out.println(stack.pop());
48
49      }
50  }
51
```

The Output

```
MATIKA\PEMROGRAMA
ericStack }
3
2
1
```