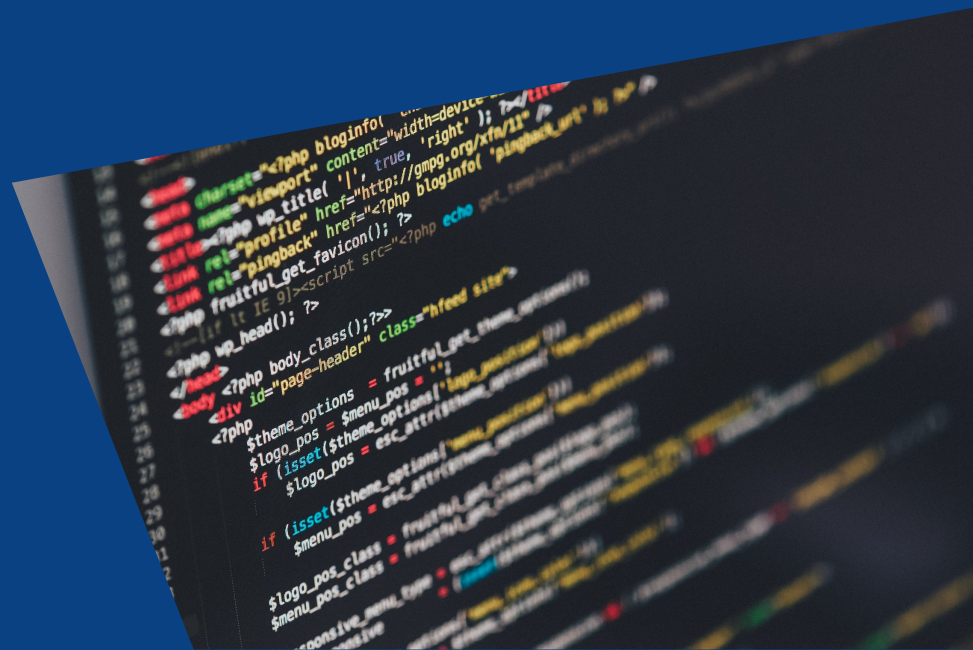
THE UNIVERSITY OF MELBOURNE

COMP90041 Programming and Software Development

Tutorial 11 Generics

Chuang(Frank) Wang



THE UNIVERSITY OF MELBOURNE

Overview

1. `ArrayList`
2. Generics
3. Exercise

THE UNIVERSITY OF
MELBOURNE

1. ArrayList

Motivation

```
public static void main(String[] args) {
    String array[] = {"hello", "there"};
    printOutArray(array);
}
```

```
public static void main(String[] args) {
    Student s1 = new Student(1, "frank");
    Student s2 = new Student(2, "mark");
    Student[] array = {s1, s2};
    printOutArray(array);
}
```

```
public static void printOutArray(String[] array){
    for (int i = 0; i < 2; i++) {
        System.out.println(array[i]);
    }
}
```

```
i < array.length
```

```
public static <T> void printOutArray(T[] array){
    for (int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}
```

ArrayList class

- ✓ `ArrayList` class is a class in the standard `java.util` library.
- ✓ `Array VS ArrayList`

  `Array` – fixed length

  `ArrayList` – its size can grow and shrink as needed during runtime

- ✓ In general, an ArrayList serves the same purpose as an array, except that an ArrayList can change length while the program is running.

Using ArrayList

```
public static void main(String[] args) {
    Student s1 = new Student(1, "frank");
    Student s2 = new Student(2, "mark");

    List<Student> list = new ArrayList<>();
    list.add(s1);
    list.add(s2);

    System.out.println(list.size());

    System.out.println(list.contains(s1));

    list.remove(s2);
    System.out.println(list.size());


}
```

1. **Create an `ArrayList` Object**
   - in the same way as object of any class
   - except that you specify the type parameter
   - type parameter defines the type of objects stored

2. `list.add(s1);`
   - Adds the specified element to the end of the calling ArrayList
   - and increases the ArrayList's size by one
   - The capacity of the ArrayList is increased if that is required.

3. `list.size();`

Returns the number of elements in the calling ArrayList.

4. `list.contains();`

Returns true if the calling ArrayList contains target; otherwise, returns false.

More details are available at page 803 <Absolute Java> edition 6

Demo

THE UNIVERSITY OF
MELBOURNE

2. Generics

Generics

- ✓ Generic Programming means writing code that can be reused for object of many different types

- ✓ Generic Solution – type parameters

```
List<Student> list = new ArrayList<>();
```

- ✓ The type parameter can be replaced by any **reference** type

Generic class (parameterized class)

- ✓ A class with one or more type parameter
- ✓ Type parameters are introduced after class name, enclosed by angle brackets <>

```
public class Pair<T>{
    ...
}
```

- ✓ Type parameters are used throughout the class definition
- ✓ The generic class is instantiated once the type parameter is specified by any reference type

```
Pair<String> secretPair = new Pair<>("Happy", "Day");
```

- ✓ Generic classes act as a factory for producing ordinary classes.

PITFALLS

- ✓ A Generic Constructor Name Has No Type Parameter

```
public Pair<T>() {
   ...
}
```

```
public Pair() {
   ...
}
```

- ✓ A Primitive Type Cannot be Plugged in for a Type Parameter

Using a generic class

Demo

Generic Method

- ✓ The type parameter must be placed (in angle brackets) after all the modifiers, and before the returned type

```
public static <T> void printOutArray(T[] array){
      for (int i = 0; i < array.length; i++) {
          System.out.println(array[i]);
      }
}
```

THE UNIVERSITY OF
MELBOURNE

3. Exercise

Tutorial Exercise

1. Write a Java program that allows users to enter and edit a list of your friends' names. The program should have a main loop that prints out the full current list of names, one per line, with each name preceded by its numeric position in the list, and prompts them for what they want to do. Give them the following options:

   - Enter names, adding them to the end of the current list. In this mode, the program keeps reading names, one per line, until the user enters a blank line.
   - Delete a name. This prompts for the position number of the name to delete.
   - Quit.

   The list of names should start out empty.

   Hint: this is a job for the `ArrayList` class.

Homework Exercise

2. Extend the program above to provide a command to move a name to a different position in the list.

3. Extend the program above to support the concept of an "insertion point", a point between two list entries (or before the first or after the last) where newly added names are placed. Add a command to your main loop to allow the user to set the insertion point, and make sure to show the insertion point when showing the list of names.

   For this part, you should create your own class, similar to `ArrayList`, that supports the concept of an insertion point. Since the idea of a list with an insertion point is flexible (it could be useful for lists of *anything*, not just strings), make this new class generic. Feel free to use the `ArrayList` class in your implementation.

THE UNIVERSITY OF MELBOURNE

Thank you