Unit 07: Generics

Anthony Estey

CSC 115: Fundamentals of Programming II

University of Victoria

Unit 07 Overview

- Related Reading:
  - Textbook Section 9.3
- Learning Objectives: (You should be able to…)
  - understand what Java Generics are, and the reasons why they are used
  - read and write Java code that uses Generics

Generics - Motivation

- Let's look at some of the **Node** classes we have used so far:

```
public class Node {
    private int data;
    protected Node next;

    public Node() { … }

    /* Purpose: get the Node's data value
     * Returns: (int) the data value */
    public int getData() {
        return data;
    }
```

```
public class Node {
    private String data;
    protected Node next;

    public Node() { … }

    /* Purpose: get the Node's data value
     * Returns: (String) the data value */
    public String getData() {
        return data;
    }
```

Only difference: **int** type vs **String** type

Generics - Motivation

- What about our List classes?

```
public interface IntegerList {
    void addFront (int val);
    void addBack (int val);
    int size ();
    int get (int position);
}
```

```
public interface StringList {
    void addFront (String val);
    void addBack (String val);
    int size ();
    String get (int position);
}
```

The operations for each method would be the same, only the type of data associated with each list element is different

Generics - Motivation

- What about if we want to make a list of **Student** objects, or **Songs**?
- It doesn't make sense to duplicate almost all of the code in the Node and List classes every time the type of data changes
- There must be a way we can program our lists so that they are more…

Generic

Generics in Java

- Generics allow the development of classes and interfaces without needing to decide **types** until the class is actually being used

- How is this done?
- Definition of the class or interface is followed by $< T >$
  - T represents the data type that the client code will specify
  - this could be a Student object, a Song object, an int, a String, etc.

Generics

- General form:

```
public class Name<T> {
    public T variable;

    public Name(T var) {
        this.variable = var;
    }
    …
```

- To create an instance of the class:

```
Name<T> x = new Name<T>(value);
```

value will be of type T

- Specific examples:

```
public class Node<T> {
    public Node<T> next;
    public T data;

    public Node(T data, Node<T> next) {
        this.data = data;
        this.next = next;
    }
```

- To create an instance of the class:

```
Node<Integer> a = new Node<Integer>(5, null);
Node<Integer> b = new Node<Integer>(7, a);

Node<String> c = new Node<String>("A", null);
```