CSE 11
Accelerated Intro to Programming
Discussion Section 8

Ayon Biswas, Spring 2021

This discussion is being recorded

Logistics

- PA7 due today at 11:59PM
- PA8 released

Generic Methods

- Enable us to run the same code for multiple data types
- we use <> to specify parameter types in generic method and class creation
- The type parameter's name is our choice, such as T, U

Example : generic method to print elements in array

```
public class GenericMethodTest {
  public static < E > void printArray( E[] inputArray ) {
    for(E element : inputArray) {
      System.out.printf("%s ", element);
    }
   }
  public static void main(String args[]) {
    // Create arrays of Integer, Double and Character
    Integer[] intArray = { 1, 2, 3, 4, 5 };
    Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
    Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

    printArray(intArray);   // pass an Integer array
    printArray(doubleArray);   // pass a Double array
    printArray(charArray);   // pass a Character array
  }
}
```

Generic Interfaces and Classes

- write it in front of the name of the interface or class instead

```
interface Checker<T> {
  boolean check(T t); // We can use T here.
  }
```

- notable difference between generic methods and classes is that when using generic classes/interfaces, Java won't figure out the actual types for us, and we need to provide them manually

```
class LongTweet implements Checker<TextTweet> { // We need to provide the TextTweet type here ...
    public boolean check(TextTweet t) {
        return t.content.length() > 20;
    }
}
```

Lists and ArrayLists

- ArrayList is an implementation of List interface provided by Java
- we used regular arrays to store lists of data, read them, and modify them. ArrayLists are a generic type in Java that allows us to do all the same things, with slightly different syntax:

int[] numArray = {1, 2, 3};	List<Integer> numList = Arrays.asList(1, 2, 3);
int one = numArray[0];	int anotherOne = numList.get(0);
numArray[1] = one * 2;	numList.set(1, anotherOne * 2);
for (int num: numArray) { /* ... */}	for (Integer num: numList) { /* ... */}

Lists and ArrayLists

- The key distinction between arrays and ArrayLists is that ArrayLists are resizable, so we can add and remove values from them

```
List<String> messages = Arrays.asList("Hello", "CSE11");

// Appends "on your assignments!" to the end of messages
messages.add("on your assignments!");

// Inserts "Good luck" at index 2, shifting "on your assignments!" to the right.
messages.add(2, "Good luck");

// Removes "Hello" from the list, and returns it.
String hello = messages.remove(0);
```

Overloading

- providing multiple implementations of the same method, differentiated by the parameters they accept

```
int add(int a, int b) { /* ... */}
double add(double a, double b) { /* ... */}
```

- only works if the different method have a different number of parameters, or parameters of different types , but not if they only have different return types
- A common use for overloading methods is to provide default parameters. For instance, the following method is overloaded to use a default length of 5

```
/*
   This method returns a new ArrayList containing the Strings in `strs` that
   are longer than `length`.
*/
static ArrayList<String> longStrings(ArrayList<String> strs, int length) {
...
}
static ArrayList<String> longStrings(ArrayList<String> strs) {
   return longStrings(strs, 5);
    }
```

Exceptions

- Java's way of handling errors that can happen when we run our programs
- run into exceptions as a result of incorrect/buggy code as well. Some of these exceptions include:
  - ArrayOutOfBoundsException: Happens when we try to get the element at an index of an array that doesn't exist
  - NullPointerException: Happens when we try to read a field of, or call a method on a null value.
  - ArithmeticException: Happens when we try to run an invalid arithmetic operation (e.g. divide by zero)
- we can provide a custom, more descriptive, error message while throwing exceptions

```
static Integer max(ArrayList<Integer> elements) {
  if (elements.size() == 0) {
    throw new IllegalArgumentException("max got an empty list");
  }

  /* ... */
  }
```

PA8

- Start early!

Thanks!