Computer Science Introductory Course MSc - Introduction to Java

Lecture 3: Exceptions, Generics, Collections

Pablo Oliveira <pablo@sifflez.org>

ENST

Outline

Exceptions

Definition

An exception is an event that indicates an abnormal condition disrupting the normal flow of the program.

- Exception is a subclass of java.lang.Throwable.
- When an exception is thrown the program is interrupted, the exception is propagated rewinding the call stack until an appropriate exception handler is found.

Creating and throwing an Exception

```
class MyException extends Exception {
  MyException() {}
  MyException(String message) {}
}

throw new MyException();
```

Catching an Exception

```
try{
  // code that can throw an exception
}
catch (MyException1 e) {
  // treatment in case of Exception1
}
catch (MyException2 e) {
  // treatment in case of Exception2
}
finally {
  // always execute at the end of the try block
}
```

- finally is optional.

Exception rules

- If a method is thrown inside a method, one these two must be true :
  - the exception is catched inside a try/catch block.
  - the method is declared to throw the exception (using keyword `throws`)
- One can throw an exception inside a catch block.
- One can catch a group of exceptions using an Exception superclass.

Example

```
class Example {
  void foo() throws MyException {
    try {
      throw new MyException();
    } catch (MyException e) {
      throw e;
    }
  }
  void bar() {
    try {
      foo();
    } catch(Exception e) {
      System.out.println("Problem!");
    }
  }
}
```


Outline

Generics

Definition

Generics are a way of instantiating classes or calling methods with a variable type paramater. This allows to remove unnecessary casts and make safer code, because the compiler knows which type he is supposed to work with and thus can generate the appropriate checks.

List without generics

Q : We want to implement a list of objects ?

```
class List {
  private static final int maxSize = 100;
  private int last = 0;
  private Object list[];
  List() {
    list = new Object[maxSize];
  }
  void add(Object o) throws Exception{
    if (last >= maxSize)
      throw new Exception("no place left");
    list[last++] = o;
  }
  Object get(int index) {
    return list[index];
  }
}

List l = new List();
l.add("Hello");
String s = (String) l.get(0);
Integer i = (Integer) l.get(0); // error at runtime!
```

List with generics

```
class List<T> {
  private static final maxSize = 100;
  private int last = 0;
  private T list[];
  List() {
    list = (T[]) new Object[maxSize];
  }
  void add(T o) throws Exception{
    if (last >= maxSize)
      throw new Exception("no place left");
    list[last++] = o;
  }
  T get(int index) {
    return list[index];
  }
}

List<String> l = new List<String>();
l.add("Hello");
String s = l.get(0);
Integer i = l.get(0); // detected at compile time!
```


Generic methods and constructors

```
class Example {
  <T> Example(T something) {}
  <T> void foo(T something) {}
}

  Example e = new Example("Hello"); //type inference,
  e.foo(new Integer(3));            //no need to pass <>
```

Outline

Collections

Definition

Collections framework is a group of classes in java libraries which eases the manipulation of group of objects.

- All the classes in the collection framework implement the Collection interface.
- The collection framework provides List, Set, Queue, Map, etc...
- The collection classes can be provided with a generic type !

Example

```
List<String> myList = new ArrayList<String>();
myList.add("Hello");
myList.add("Bye");
myList.get(1);
java.util.Collection.sort(myList);
```


The for-each control construct

```
List<String> myList = new ArrayList<String>();
for (String s : myList) {
  System.out.println(s);
}
```