Java training

Generics

Session overview

- **Generics**
- **Hands-on** - using & testing each sub-topic

Object type system

- All Java classes extend the `Object` class
  - They inherit all the methods from it
- Default type for all the collections & arrays of items
- Problem:

```
List list = new List();      // not typed, may contain apples and / or lions
list.add("something");
list.add(2);
```

Generics

- Introduced by Java 5.0 (JDK 1.5)
- Allow / enforce specifying types for collections / arrays of elements
- The previous problem becomes:

```
List<Product> products = new List<Product>(); // 'list of products'
products.add("something");                    // compile time error
```

Generic classes

- Generics can also be applied at class level

```
public abstract class Stack<Element> {     // Element → generic type

}
```

'type placeholder'

- The usage of '<' and '>' - called ***'type placeholder'***
- When used, they define several *generic types* that can be used by that class
- A class can use any number of generic types

Example

```
public abstract class Stack<Element> {
    // methods that implement a FIFO stack
}
```

Applied to a real class:

```
public class ProductStack extends Stack<Product> {
    // the methods will now work on a defined type, not on a generic type
}
```

Second example

```
interface Converter<Input, Output> {
    // methods to convert Input → Output and Output → Input
}
```

An implementation of the Product class:

```
class ProdConverter implements Converter<Product, AnotherProduct> {
    // methods for the specific types Product (as Input) & AnotherProduct (as Output)
}
```

Generics for wildcards and class hierarchies

- Generics can also be applied to:
  - Class hierarchies - using *extends*
  - Wildcards - using *?*
- Example:

```
public abstract ProductProcessor<T extends Product> {
    public void process(T t);
}
List<?> items = query.listItems();
    // for methods with an unknown / generic result type
```

Diamond operator

- **'Diamond' operator - '<>'** (since Java 7)
- Used when the compiler can infer / deduce the argument types
- Example:

```
Queue<Product> productsQueue = new Queue<Product>(10);
```

Can be written as:

```
Queue<Product> productsQueue = new Queue<>(10);
```

Generic methods

- Methods that use generic parameter types
  - Declare generic methods in an interface / class even if itself is not generic
- Mainly used as utility methods
  - Example classes: Collections, Arrays, ...
- Example:

```
public static <T> void display(List<T> items, int until) {
    for (int i = 0; i < until; i++) { // ... perform actions
    }
}
```

Hands-on

- Implement a ‘displayer’ class for a hierarchy of Product classes
  - Just display: ‘The displayed product is ‘ and then call the object’s #toString
- Implement a class which holds a tuple (pair) of objects, of different types (T1, T2)
- Try an implementation of a generic class with incorrect parameters → get used to the error messages
  - Examples:
    - Use a single generic parameter for a class defined for two parameters
    - Use an incorrect parameter for a generic class