Lab: Generics

This document defines the lab for "Java Advanced" course @ Software University. Please submit your solutions (source code) of all below described problems in Judge.

1. Jar of T

Create a class **`Jar<>`** that can store **anything**.

It should have two public methods:

- **`void add(element)`**
- **`element remove()`**

Adding should add on **top** of its contents. Remove should get the **topmost** element.

Examples

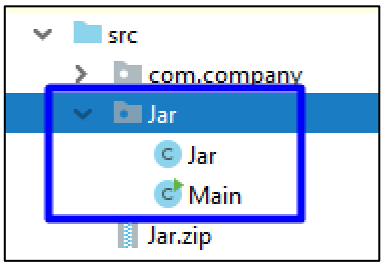
```
Jar<Pickle> jarOfPickles = new Jar<>();

jarOfPickles.add(new Pickle());
jarOfPickles.add(new Pickle());

Pickle pickle = jarOfPickles.remove();
```

Hints

Use the syntax **`Jar<T>`** to create a generic class

Submit in Judge

Submit your solution in Judge **zip** your whole package with the **`Jar`** and **`Main`** **classes**:



If you didn't create **package** just choose your classes and **zip** them.

2. Generic Array Creator

Create a class **`ArrayCreator`** with a method and a single overload to it:

- **`static T[] create(int length, T item)`**
- **`static T[] create(Class<T> class, int length, T item)`**

The method should return an array with the given length and every element should be set to the given default item.

Examples

```
String[] strings = ArrayCreator.create(10, "none");
Integer[] integers = ArrayCreator.create(Integer.class, 10, 0);
```

3. Generic Scale

Create a class **`Scale<T>`** that holds two elements - **left** and **right**. The scale should receive the elements through its single constructor:

- **`Scale(T left, T right)`**

The scale should have a single method:

- **`T getHeavier()`**

The **greater** of the two elements is heavier. The method should return **`null`** if elements are **equal**.

Examples

```
Scale<String> stringScale = new Scale<>("a", "c");
System.out.println(stringScale.getHeavier());

Scale<Integer> integerScale = new Scale<>(1, 2);
System.out.println(integerScale.getHeavier());
```

4. List Utilities

Create a class **`ListUtils`** that you will use through several other exercises:

The class should have two static methods:

- **`T getMin(List<T> list)`**
- **`T getMax(List<T> list)`**

The methods should throw **`IllegalArgumentException`** if an empty list is passed.

Examples

```
List<Integer> integers = new ArrayList<>();
Collections.addAll(integers, 1, 2, 18, 2, -1);

Integer maxInteger = ListUtils.getMax(integers);

List<String> strings = new ArrayList<>();
Collections.addAll(strings, "a", "b", "c");

String minString = ListUtils.getMin(strings);
```