Java - elements of generic programming (I)

Working environment setup

1. Download and unzip `lab04` source code
    1. Download `lab04.zip` from the course site (moodle)
    2. Unzip it (you get `lab04` directory)
    3. Move `lab04` to `programming-in-java` directory, i.e.,
        - `programming-in-java`
            - `lab00`
            - `lab01`
            - `lab02`
            - `lab03`
            - `lab04 <--`
            - `gradle`
            - ...
2. [`IntelliJ`] Add `lab04` module to the `programming-in-java` project
    1. In the *Project* window click `settings.gradle` file to open it
    2. Modify its content to the following:

```
rootProject.name = 'programming-in-java'
include 'lab00'
include 'lab01'
include 'lab02'
include 'lab03'
include 'lab04'
```

    3. Save the file
    4. Click `Load Gradle Changes` (a small box in the top right corner)

1) Concepts of parametric polymorphism, type constructor, and type variable

Exercises

1. Familiarise yourself with The Java Tutorials > Generics
2. Look briefly at the chapters of `Java Language Specification` related to:
    - `Generic Classes`
    - `Generic Methods`
    - `Generic Interfaces`

2) Generic methods, classes, and interfaces

Analyse the source code in packages:

- `lst04_01`
- `lst04_02`
- `lst04_03`

Exercises

1. Explain the benefits of using generic types
2. Explain the syntax of:
    - generic class declaration
    - generic method declaration
3. Explain what a *raw type* is, why it is unsafe, and why the *raw types* are allowed in Java
4. Given `GenBox` as defined in `lst04_01` explain the compilation result of:

```java
// (a)
GenBox gb1 = new GenBox(1);
gb1.setX("abc");
gb1.setX(new GenBox(true));
```

```java
// (b)
GenBox<Integer> gb2 = new GenBox(1);
gb2.setX("abc");
```

```
gb2.setX(new GenBox(true));
```

5. [c] Complete the method header in the following code so that it compiles:

```
public class Main {
    public static void main(String[] args ) {
        Integer[] ints = {1, 2, 3};
        String[] strs = {"A", "B", "C"};

        print(ints);
        print(strs);
    }

    ____________________________________ { // <- complete this
line
    for (int i = 0; i < elems.length; i++)
        System.out.print(elems[i] + " ");
    System.out.println();
    }
}
```

6. [c] Refactor the source code to `one file-one class` structure

7. [c] Implement the generic class `Pair<F,S>` (see `exc04_01`):
   - add at least one constructor (two parameters: `F fst` and `S snd`)
   - add the accessors ("getters") and mutators ("setters")
   - add `toString` , `equals` , `hashCode`
   - add `clone` method
   - add unit tests

3) Bounds for type variables

Analyse the source code in packages:

- `lst04_04`
- `lst04_05`

Exercises

1. Explain the purpose of bounds for type variables
2. Check if a type variable may have many interface bounds. Repeat this for class bounds.
3. [c] Change the following generic function so that it compiles

```
// Moving all elements of the array to point (x,y)
private static <T> void moveAll(T[] elems, double x, double y) {
    for (var e : elems) {
        e.goTo(x, y);
    }
}
```

Hint: first declare interface `Moveable` , and then use it as the bound for the type variable

4) Subtyping and Wildcards

Analyse the source code in package `lst04_06`

Exercises

1. Explain the notions of

   - *invariance*
   - *covariance*

   - *contravariance*

   of generic types (type constructors)

2. Explain the notions of:

   - *subtype wildcard*
   - *supertype wildcard*
   - *unbounded wildcard*

3. Given:

```
class A {}
class B extends A {}
class C extends B {}
class GenBox<T> {
    private T x;
    public T getX() { return x; }
    public void setX(T x) { this.x = x; }
    //...
}
```

from the following lines point out these that do not compile (explain each error):

```
GenBox<B> gb1 = new GenBox<B>();
GenBox<B> gb2 = new GenBox<C>();
GenBox<B> gb3 = new GenBox<A>();
B b1 = gb1.getX();
gb1.setX(new B());

GenBox<? extends B> gb4 = new GenBox<B>();
GenBox<? extends B> gb5 = new GenBox<C>();
GenBox<? extends B> gb6 = new GenBox<A>();
B b2 = gb5.getX();
gb5.setX(new B());
gb5.setX(new C());

GenBox<? super B> gb7 = new GenBox<B>();
GenBox<? super B> gb8 = new GenBox<C>();
GenBox<? super B> gb9 = new GenBox<A>();
B b3 = gb9.getX();
gb9.setX(new B());
gb9.setX(new C());

GenBox<?> gb10 = new GenBox<B>();
GenBox<?> gb11 = new GenBox<C>();
GenBox<?> gb12 = new GenBox<A>();
B b4 = gb10.getX();
gb10.setX(new B());
```

5) Mini project 04_01 (`exc04_02`)

[c] Implementation of generic interfaces `MyStack<E>` and `MyQueue<E>` :

1. Augment these interfaces with exception handling
2. Complete `MyStackDLLBImpl` and `MyQueueDLLBImpl` (DLLB - Doubly Linked List Based)
3. Add JavaDoc comments to both interfaces and implementation classes
4. Write unit tests

6) Push the commits to the remote repository