TYPE SAFETY AND CODE REUSE : USING GENERICS

NOW LET'S SAY YOU NEEDED A CLASS TO HOLD 3 STRINGS

THIS IS PRETTY SIMPLE TOO, NOW THE QUESTION IS HOW BEST TO DO THIS

OPTION 1: REWRITE THE CLASS WE JUST WROTE, BUT REPLACE EVERY "DOUBLE" WITH 'STRING'

OPTION 2: MODIFY THE CLASS WE JUST WROTE, REPLACE "DOUBLE" WITH OBJECT, SO THAT THE SAME CLASS WILL WORK FOR BOTH NUMBERS AND STRINGS

NOW LET'S SAY YOU NEEDED A CLASS TO HOLD 3 STRINGS

THIS IS PRETTY SIMPLE TOO, NOW THE QUESTION IS HOW BEST TO DO THIS

OPTION 1: REWRITE THE CLASS WE JUST WROTE, BUT REPLACE EVERY "DOUBLE" WITH 'STRING'

OPTION 2: MODIFY THE CLASS WE JUST WROTE, REPLACE "DOUBLE" WITH OBJECT, SO THAT THE SAME CLASS WILL WORK FOR BOTH NUMBERS AND STRINGS

(THIS WOULD WORK BECAUSE ALL REFERENCE TYPES DERIVE FROM OBJECT, AND WE WOULD JUST USE THE DOUBLE CLASS INSTEAD OF THE DOUBLE PRIMITIVE TYPE)

GENERIC CLASSES WERE BUILT FOR EXACTLY THIS TYPE OF SITUATION!

OPTION 1 IS A BAD IDEA - COPY-PASTING CODE ALWAYS IS

OPTION 2 IS A BAD IDEA TOO - ITS NOT TYPE-SAFE - WHAT IF YOU CALLED IT WITH 2 NUMBERS AND 1 BOOLEAN BY MISTAKE?

JAVA LET'S YOU WRITE A GENERIC TRIPLE CLASS, WHERE THE EXACT TYPE OF THE DATA TO BE HELD IS SPECIFIED BY THE USER WHEN AN OBJECT OF THE CLASS IS INSTANTIATED

JUST ADD A TEMPLATE PARAMETER TO THE NAME OF THE CLASS, AND THEN REPLACE EVERY 'DOUBLE' WITH THE TEMPLATE PARAMETER

CREATING A GENERIC CLASS

```
public class Triple<T> {

    private T first;
    private T second;
    private T third;

    public T getSecond() {
        return second;
    }

    public void setSecond(T second)
        this.second = second;
    }

    public T getThird() {
        return third;
    }
```

NOTICE THE <T>, CALLED THE TYPE PARAMETER

IN THE BODY OF THE CLASS SIMPLY REPLACE EVERY SPECIFIC TYPE WITH THE GENERIC TYPE T

INSTANTIATING A GENERIC CLASS

```
Triple<Double> threeNumbers = new Triple<Double>();
Triple<String> threeStrings = new Triple<String>();
```

SIMPLY INSTANTIATE THE CLASS WITH THE CORRECT TYPE SPECIFIED IN <ANGLE BRACKETS>

BTW, WHAT IF YOU HAD WANTED THE THREE VARIABLES OF DIFFERENT TYPES? WOULD THAT HAVE BEEN POSSIBLE?

CERTAINLY, JUST HAVE 3 TEMPLATE PARAMETERS INSTEAD OF 1

A GENERIC TRIPLE WITH 3 DIFFERENT TYPES WOULD LOOK LIKE THIS

IT WOULD NOW NEED NOT 1 BUT 3 TYPE PARAMETERS TO INSTANTIATE AN OBJECT OF THIS CLASS

```
Triple<Double,String,Integer> threeValuesOfDifferentTypes = new Triple<Double,String,Integer>();
```

```
public class Triple<T1, T2, T3> {
    private T1 first;
    private T2 second;
    private T3 third;

    public T2 getSecond() {
        return second;
    }

    public void setSecond(T2 second) {
        this.second = second;
    }

    public T3 getThird() {
        return third;
    }

    public void setThird(T3 third) {
        this.third = third;
    }

    public T1 getFirst() {
        return first;
    }

    public void setFirst(T1 first) {
        this.first = first;
```

WE PREVENT CODE DUPLICATION WHILE STILL GETTING TYPE SAFETY

THIS LIES AT THE HEART OF

GENERIC COLLECTIONS