**1- short + short is not equal to short**

The code above looks simple with no visible errors, but in fact, it does not compile.

The compilation error is “*The method returnShort() is not applicable for the argument int.*” the reason for that is because in java, **the sum of two shorts is an int**.

This is not restricted to shorts, in fact **every type that is under int will be promoted to int**.

This is called the [binary numeric promotion](https://docs.oracle.com/javase/specs/jls/se8/html/jls-5.html#jls-5.6.2).

**2- ArrayList size vs ArrayList capacity**

In this example we initialize a list of Integer with a capacity equal to 3 then we add 4 elements to the list. What is the expected behavior ?

The code compiles without any errors and for the output “*size 1 : 0 size 2 : 4*” is displayed.

The ArrayList class has among others two properties:

-**Size** : The number of elements in the list.

-**Capacity** : The number of elements that the list can contain without re-allocating its internal structure.

In our example we used an ArrayList [constructor](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#ArrayList%28int%29) that constructs an empty list with the specified initial capacity, then it expands if we add a number of elements that are greater than the initial capacity.

**3- Two strings with the same value can not be equal.**

Two strings that have the same value are not equals because they do not reference the same element (String Interning).

Thanks to the immutability of Strings in Java, the JVM can optimize the amount of memory allocated for them by storing only one copy of each literal String in the pool. This process is called **interning**.

When we create a String variable and assign a value to it, the JVM searches the pool for a String of equal value.

If found, the Java compiler will simply return a reference to its memory address, without allocating additional memory.

If not found, it’ll be added to the pool (interned) and its reference will be returned.

**4- Should we set a no args constructor or not ?**

Graphical user interface, application, Word

Description automatically generated

If no constructor is set in a class then java creates a default constructor, but If we set a constructor with one or more parameters then the default constructor will not be generated and we must set it manually if we need it.

So the rule is “**If you declare one or more constructors, Java will NOT create an implicit noargs constructor for your.**”

**5- The isA test.**

Text, application

Description automatically generated

The float value is not casted to Double because float and Double are completely different; float is a primitive type and Double is a wrapper class. So the float value **does not meet the isA test** for Double.

Graphical user interface, text, application

Description automatically generated

List of Object does not accept List of Integer although Integer is an Object.

List <Object> only accepts a List which contains Object, the List <Integer> **does not meet the isA test** for List<Object> and would not be passed to a List <Object> parameter.

This is a common misunderstanding when it comes to programming with generics, but it is an important concept to learn. You can take a look at the [java specification for generics](https://docs.oracle.com/javase/tutorial/java/generics/inheritance.html) to get all the details about this topic.

**6- Default access modifier for reference types.**

**If no access modifier is set to a reference type (class, enum or interface) then it has the ‘package-private’ access by default.** That means it can be accessed only inside the package where it is implemented (it is not private or public). You can find an example [here](https://github.com/NoorKrichen/7-tricky-parts/tree/main/src/trick6).

**7- HashSet can be equal to LinkedHashSet although they are two different objects**

HashSet and LinkedHashSet extend the class AbstractSet so they inherit the method [equals](https://docs.oracle.com/javase/7/docs/api/java/util/AbstractSet.html#equals%28java.lang.Object%29) from it. In the official implementation we find that this method returns true if the given object is also a set, the two sets have the same size, and every member of the given set is contained in this set. This ensures that the equals method works properly across different implementations of the Set interface. **In general we must look to the implementation of the used equals method.**