## Aufgabe 1

Explain Upcasting and Downcasting and write a Code Example  
Explain it with Objects and Primitive Datatypes

Upcasting: Means casting a variable or its value into one of bigger type the smaller will fit in in any way, as long as there are logically or hierarchically compatible.

For example, you can upcast a short into an integer (primitive) but also a Short into an Integer (Object), ore more practical: A Cat can also be considered as an animal, but not every animal is a cat. This by the way can be done also implicitly because there won’t be data loss. Otherwise upcasting might make a reference and its function less specified.

So Upcasting makes objects from the same root collectable under one hood.

Downcasting:   
Obviously it’s the opposite from above and makes possible to specify an object more specifically, if compatible. But in this case you are forced to downcast explicitly so you must consider possible data loss. So if cat and dog objects are collected in a list of (abstract) animals, you won’t make an animal itself bark or meow, because the compiler doesn’t know if the real object behind it is capable of one of these. So you either have to downcast (blindly/directly) if you are sure it well be what you expect, or you have to make sure it is an instance of the wanted type by instanceof (cat/dog). So the compiler will be sure you won’t try to make a cat bark :D

As primitives you won’t be able to fit a double into a byte, unless you do it explicitly and accept the risk of data loss, like floating points will be truncated (floor rounded) and if the number is out of range (-128 - +127, 28) you will get a NumberFormatException (or ClassCast when using Wrappers, if I got this right).

public static void main(String[] args) {  
 //Upcasting primitive  
 double d = 100f;  
 //Upcasting Object  
 Exception e = new RuntimeException(); //ClassCastException  
 Enemy enemy = new ShadowGolem(); //implicit upcast  
 Object o = new String("Hi");  
 //Integer i = (Integer) Short.valueOf((short)1); //Error  
  
 //Downcasting primitive  
 byte b = (byte) 100L; //blank input of 1 is promoted as an int,  
 // that's why f and L suffix is needed as well, same with short  
 //Downcasting Object  
 ShadowGolem golem = (ShadowGolem) enemy;  
 String s0 = (String) o; //bare casting, risky!  
 String s1 = (o instanceof String) ? (String) o : null; //explicit upcast after testing  
 String s2 = (o instanceof String string) ? string : null; //with pattern variable  
 //RuntimeException re = (RuntimeException) new Exception(); //ClassCastException  
 //Short s = (Short) Integer.valueOf(100); //Error

## Aufgabe 2

Complete the Code provided as Zip Archive