# Assignment – Java Core Days 1+ 2

## Exercise 1

Design a simple domain model (<http://en.wikipedia.org/wiki/Domain_model>) and implement the corresponding classes that represent the typical accounts used in a bank.

More specifically your design should reflect the following concepts/entities:

* Current Account (<http://en.wikipedia.org/wiki/Current_account_(banking)>)
  + Has a
    - Description
    - Account number (IBAN)
    - Balance
    - A boolean flag that indicates if it allows overdraft (<http://en.wikipedia.org/wiki/Overdraft>)
* Savings Account (<http://en.wikipedia.org/wiki/Savings_account>)
  + Has a
    - Description
    - Account number (IBAN)
    - Balance
    - Interest Rate
* Account Holder
  + Has a
    - Name
    - List (use array) of accounts (they can be current or savings accounts)

**Hint:** Avoid code (i.e. fields, methods) duplication.

Then create an AccountApp class with just a main() method that instantiates an account holder (named James Bond) with 3 accounts:

* A current account
* 2 savings accounts for each of his 2 children (Mary and John)

**Hint:** The output presented at the end of exercise 2 should provide you with more details if needed.

## Exercise 2

Add an interface called Printable to the previous design which should contain a single method with the signature:

void print();

The interface should be implemented by all the classes in your domain model (i.e.: all the entities mentioned above) and the implementation of the print() method should print to System.out the details of the entity.

For example, in the case of a current account it should print the account number, account description, balance and if overdraft is enabled.

Now, call the print() method on the account holder instantiated in the previous exercise and it should display an output similar to the following:

Name: James Bond

Account No: 111

Account Desc: JB's Personal Account

Account Balance: 1000

Overdraft Enabled: true

Account No: 222

Account Desc: John's Savings

Account Balance: 4000

Interest rate: 3.7

Account No: 333

Account Desc: Mary's Savings

Account Balance: 5000

Interest rate: 4.1

**Hint:** Avoid code duplication, again!

**Hint 2:** Think POLYMORPHISM! ☺

Create a Test class to test some of the existing functionality (the sky is the limit). Add several test methods for CurrentAccount, SavingAccount and AccountHolder.

## Exercise 3

This exercise is on **ArrayLists**.

You are to design a class called **PhoneBook** that uses an ArrayList to store a collection of contacts.

Start by designing the **Contact** class that holds a person’s last name, first name, home address and phone number.

**Note**: the **Contact** class should override the “toString()” method

**Note**: the phone number is a String which has 12 characters (**0720.123.123**)

Next design the **PhoneBook** class that uses an ArrayList of parameter type Contact:   
  
**Sample**: public class PhoneBook  
{  
private ArrayList<Contact> list;  
  
//rest of the code  
}

The **PhoneBook** class should have the following methods:

1. add a contact
2. display all contacts
3. search for a specific contact and display it
4. search and delete a specific contact.

With regards to the **search methods** consider the following: For example, if “**Ana**” (or “**ana**”) is the search target, then any contact where **Ana/ana** matches the **first name** or the l**ast name** should be displayed. Similarly, if “**0720-123**” is the search target, all contacts having this number should be displayed.

Also, pay attention that the contact names might be inserted in a non-generic manner.

For console display (**point 3**), you should use the **StringBuilder** and provide contact information as follows:

**Sample:**

“Following contact(s) matches your search:

1. ANA Lastname1 …. +40720-255-252

Address: str Palat, United Bussines Center 1, 3E, Iasi

1. ANA Lastname2 …. +40720-254-253

Address: str Palat, United Bussines Center 2, 3E, Iasi”

* The first name should be displayed with capital letters.
* The last name should have only the first letter with capital.
* The phone number should have the **+4aaaa-aaa-aaa** format

1. **Hint**: you use the **“\n”** (new line) and **“\t”** (tab space)

Create a Test class to test some of the existing functionality (the sky is the limit). Add several test methods to test the **PhoneBook** app.