Java - Inheritance

C&G criteria: 1.4.2, 1.4.4, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 3.1.1, 3.1.2, 3.1.3, 3.2.1, 3.2.2, 3.2.3, 3.3.2, 3.4.2, 3.4.3, 3.5.2, 4.2.2

A Class can be given additional functionality beyond its initial definition by extending the Class. The following Class has access to all the methods of the Java Class Applet as well as any methods that might be created for it.

```
public class myClass extends Applet { }
```

User defined Classes can be manipulated in the same fashion. A basic Class is created whose methods cover general functions. More specific Classes inherit all the methods of the original or Super Class and add their own methods or change - override - the existing methods of the Super Class. Overriding methods is not the same as overloading methods. Overloading is the declaration of more than 1 method with the same name within a single Class. The overloaded methods accept different parameters such as the 2 constructor methods of the Brick Class.

By having a relatively simple Super Class that is extended by more complex Sub Classes different Classes that perform similar functions can be created. As much of the code is shared the more complex Sub Class may be shorter than its Super Class as none of the inherited methods have to be rewritten unless they are to be overridden.

In this example the function of a bank will be modelled by Java Classes. A Super Class will handle common banking functions and Sub Classes will customise the model to represent examples of bank account types. To check that all this works another Class will be required as a user interface to the bank Classes.

Bank
balance number name
withdraw deposit setNumber setName getName getNumber getBalance

SavingsBank
interest
withdraw calcInterest

ChequeBank
fee rate
calcRate

SavingsBank pays interest on cash in the bank. It does not allow a customer to be overdrawn.
ChequeBank allows a customer to withdraw cash that is not there but imposes a fee and charges the customer interest.

- The original withdraw method of the Super Class Bank is overridden by SavingsBank as this Class does not allow withdrawals to take the balance below £0. ChequeBank uses the Super Class withdrawal method.
- CalcInterest in SavingsBank pays interest on money in credit, there are no negative balances. CalcRate in ChequeBank charges interest on negative balances. No interest is paid for being in credit. The 2 methods could have the same name as they belong

to different Classes. They are named differently to avoid calling the wrong method or referencing the wrong object.

Here is the Super Class Bank

```
public class Bank
{ //protected variables are not public – surprise – but private to a Class and all its Sub Classes
// all Sub Classes inherit a private balance, number and name.
	protected float balance;
	protected String number;//never going to add up acc numbers
	protected String name;
	public Bank(float deposit)
	{//constructors
		balance=deposit;
	}
	public Bank()
	{ // default deposit of £10
		balance = 10f;
	}
	public void withdraw(float sum)
	{//money out
		balance=balance-sum;
	}
	public void deposit(float sum)
	{//money in
		balance = balance + sum;
	}
	public void setNumber(String num)
	{//assign account number – a String
		number=num;
	}
	public void setName(String user)
	{//assign name
		name = user;
	}
	//get methods to return data
	public String getName()
	{
		return name;
	}
	public String getNumber()
	{
		return number;
	}
	public float getBalance()
	{
		return balance;
	}
}
```

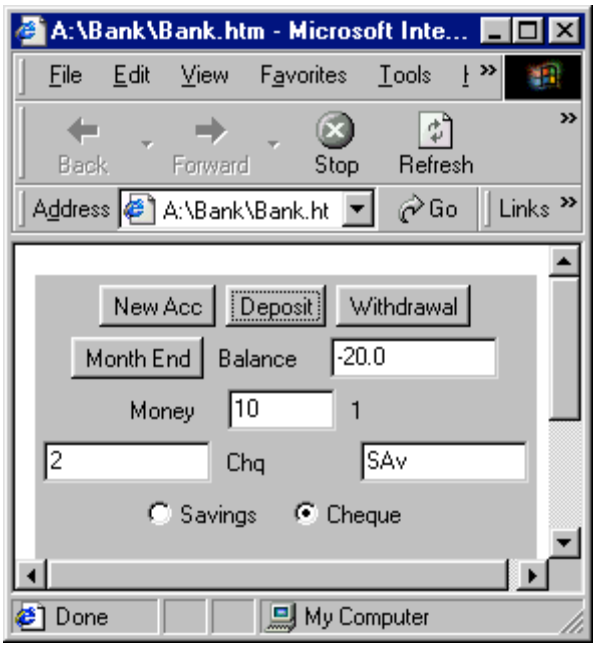
The Sub Classes inherit all the methods and variables of the Super Class that they do not override. These Classes will be of the form:

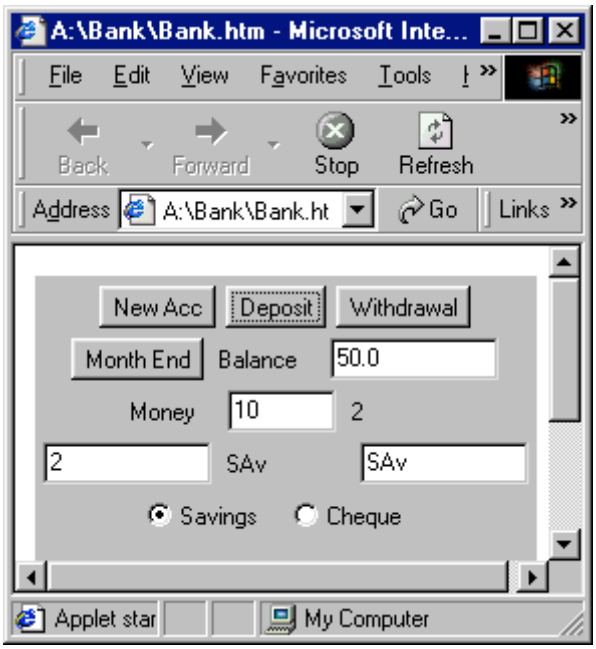
```
public class ChequeBank extends Bank
{
        //private variables
        public ChequeBank()
        {//default constructor
        }
        //public methods of the Class
        //possible private methods – only used within this Class
}
```

Implementation

1. The Bank Class can be used as it stands and possibly modified later. The 2 Sub Classes can then be created.

2. The program will require another Class to act as the user interface. It will need to be able to determine which Sub Class is being referenced by each event. Instances of the Sub Classes will have to be created with new. The following interface works but any suitable GUI can be created. There is no distinct balance Button but the user can withdraw or deposit £0 and see the balance.





3. Although the Super Class is inherited by the Sub Classes there is no specific instance (object) of the Class Bank required by this program. The Super Class can be useful when 1 of the inherited methods is called by either Sub Class. This is likely to occur when the method has not been overwritten. If a Super Class method is not used by all Sub Classes then it is likely that the method should not be in the Super Class at all and should be moved down a rung in the Class ladder.

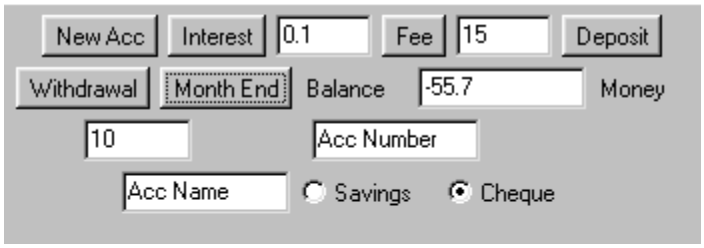
In the above interface Class the CheckboxGroup is used to determine which Sub Class to reference by assigning the name of the Sub Class in question to an instance of the Super Class.

```
public void actionPerformed(final java.awt.event.ActionEvent event)
{
        float fCash;
        Bank myAcc;
        if(chqAcc==true)//set Bank for 1 of 2 Classes
                myAcc=myBank; //object of ChequeBank
        else
                myAcc=mySave; //object of SavingsBank
        if(event.getSource()==open) // Button with caption "New Acc"
        {// call methods of Super Class
                myAcc.setNumber(accNo.getText());
                myAcc.setName(accName.getText());
        }
        //methods continue
}
```

4. There is no code outside of the constructors to change the interest paid by SavingsBank, the interest due to ChequeBank or the fee for going overdrawn each month. Create suitable methods in the appropriate Classes and adapt the interface to test these methods.

The modified Classes should be documented using Unified Modelling Language (UML) – the boxes and lines used to describe Classes, their private variables and public methods.

Class Name
Private variables
Public methods



As the interest calculations can be complex it makes sense to check them to ensure the methods are producing the expected results. Use Excel to compare expected results to those produced by the program.

Init Bal			Cheque	-£20	Savings	£20
Month	Interest	Fee	Expected Bal	Actual Bal	Expected Bal	Actual Bal
1						
2						
3						