University of HUDDERSFIELD
Inspiring tomorrow's professionals

Software Design & Development CFS2160

Week 16 – Java Inheritance

Session Plan

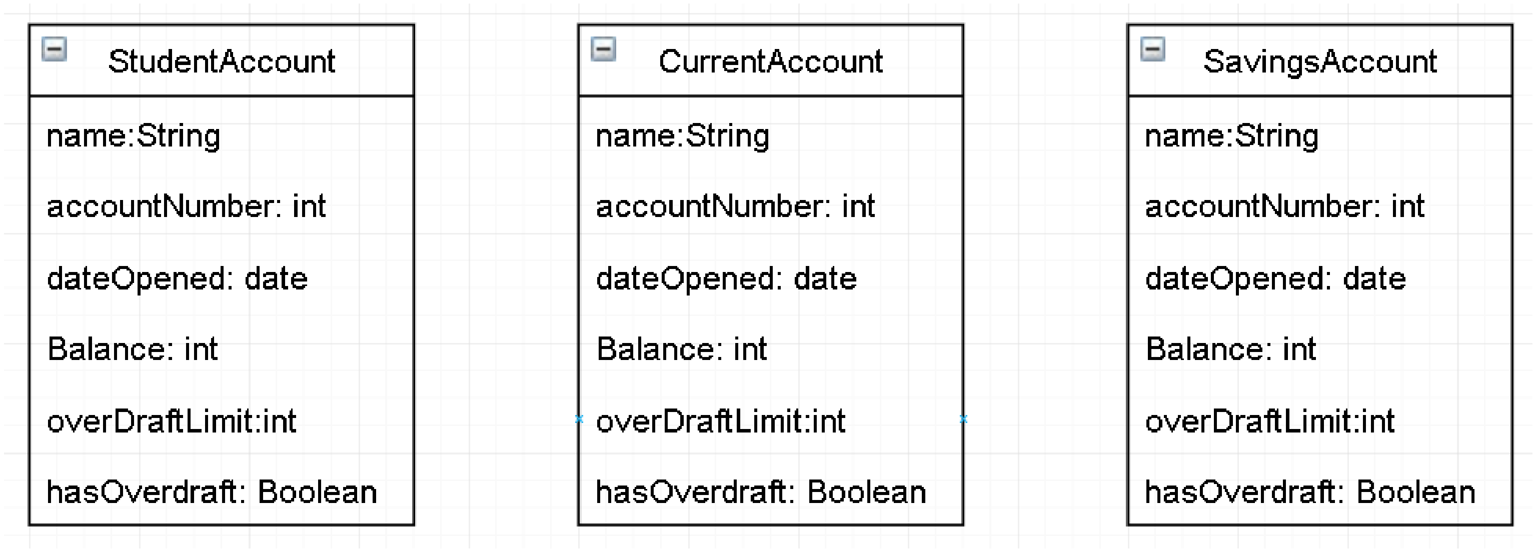
- Look at the importance and use of inheritance.
- Work through an example (start IntelliJ now please).
- Finally.

Example Classes

University of HUDDERSFIELD
Inspiring tomorrow's professionals

Lets look at the Accounts Classes in the bank systems.

What are *some* common Attributes of the classes?

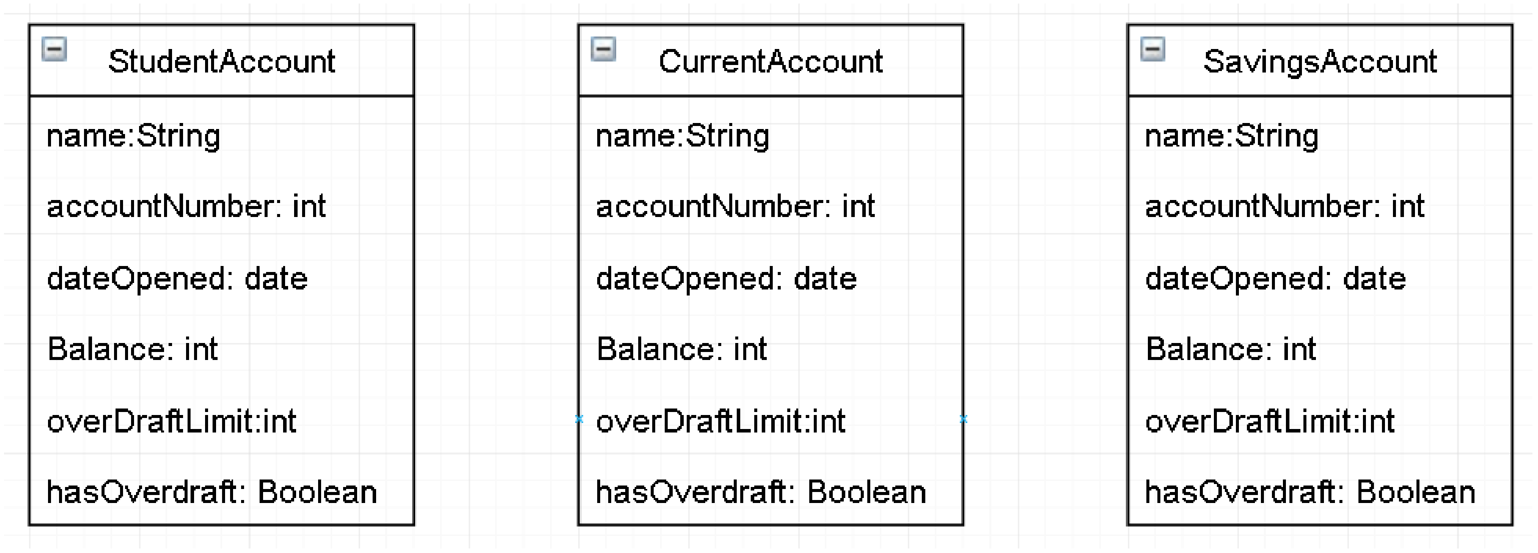


What is wrong with these classes?

Example Classes

Lets look at the Accounts Classes in the bank systems.

What are *some* common Attributes of the classes?



They have got similar attributes therefore code smell and we don't like it!

Java Inheritance

Tony always says we should whenever possible re-use code to avoid code smell.

If we were to add those common attributes into each of our account classes, we would be breaking that rule!

It would be nice if we could avoid this, we can with inheritance.

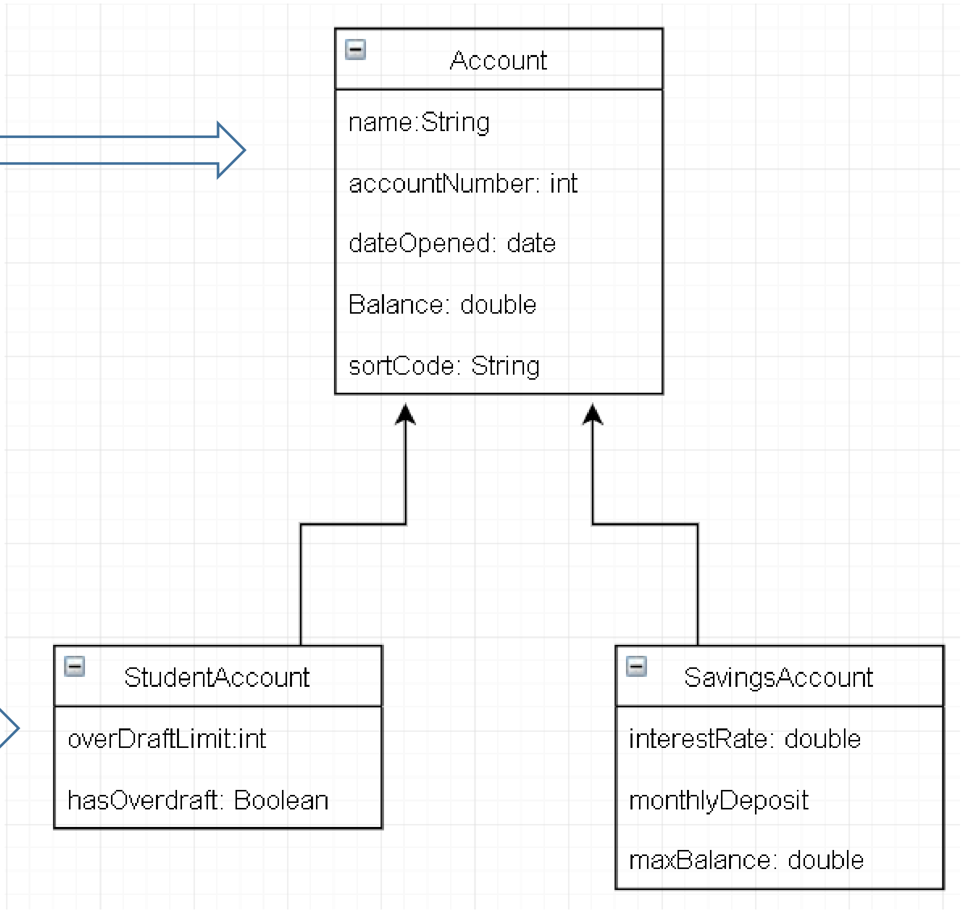
If we create an Account class we can then create child classes, such as StudentAccount that extend / inherit (therefore can use) the attributes and properties of Account.

Example Classes - Improved

University of HUDDERSFIELD
Inspiring tomorrow's professionals

We have a 'Parent Class' called Account which contains just the attributes and methods that are common to all possible types of account class

We then have 'child classes' which have unique attributes and *Extend* (inherit) the attributes of its Parent Class.



Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

- Create a new package called Bank
- Create a new class called Account in the package

```
public class Account {
    private String name;
    private int balance;
    private String accountNumber;
}
```

Add the required attributes to the class

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

- Add a constructor with all attributes to the Account class
- Add the default (empty) constructor to the Account class

```
public Account(String name, int balance, String accountNumber){
    this.name = name;
    this.balance = balance;
    this.accountNumber = accountNumber;
}
public Account(){}
```

- Add all getters and setters to the Account class
- Add withdraw() and deposit() methods to the Account class

Initial Account Class

```java
package Bank;

public class Account {
    private String name;
    private int balance;
    private String accountNumber;

    public Account(String name, int balance, String accountNumber) {
        this.name = name;
        this.balance = balance;
        this.accountNumber = accountNumber;
    }
    public Account(){}

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getBalance() {
        return balance;
    }
    public void setBalance(int balance) {
        this.balance = balance;
    }
    public String getAccountNumber() {
        return accountNumber;
    }
    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }
    public void withdraw(){}
    public void deposit(){}
}
```

You should now have a class that resembles something like this.

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

- Add a class called StudentAccount to the package
- Add 'extends Account' to the class as shown below

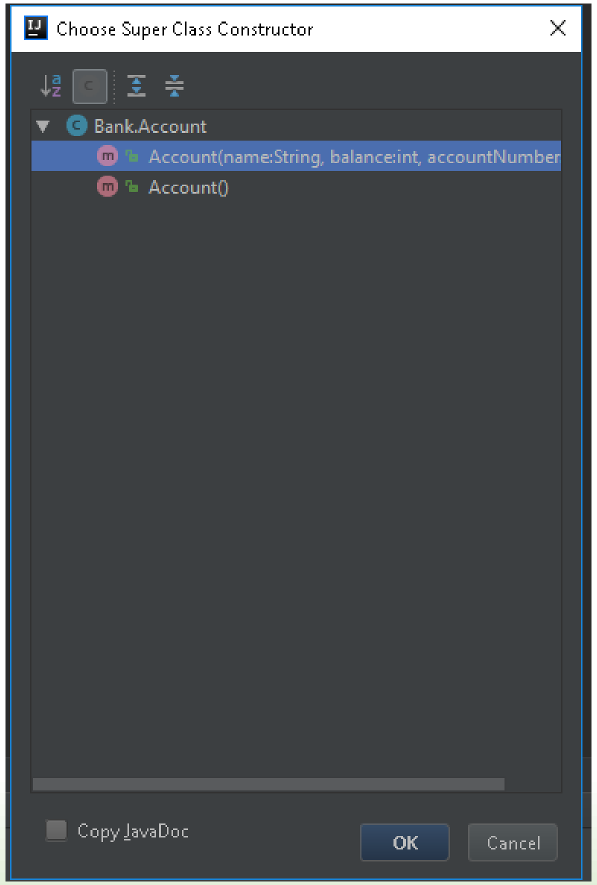
```
package Bank;

public class StudentAccount extends Account{


}
```

Extends is the keyword used to inherit a class into another

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

- Add a constructor to the StudentAccount class using 'Code>Generate>Constructor'



Choose Super Class Constructor

Bank.Account
Account(name:String, balance:int, accountNumber
Account()

We now have the option to choose which constructor in the super class (Account) we wish to use, select the one with the values in the round brackets.

Note, In the title bar the message says 'Choose Super Class Constructor'. This means we are using the constructor of the super class (Account).

Inheritance in Action

Our StudentAccount class should now look like this

```
package Bank;

public class StudentAccount extends Account{
    public StudentAccount(String name, int balance, String accountNumber) {
        super(name, balance, accountNumber);
    }
}
```

You can see that although our StudentAccount class does not have attributes, the ‘Code>Generate>Constructor’ function of intellJ has added *name*, *balance* and *accountNumber* to the constructor, these attribute have been fetched from the Account (super) class constructor, this can happen because we added the ‘extends’ keyword and the ‘Account’ class name to StudentAccount.

The ‘super’ keyword is used to pass the values gathered when creating an object of StudentAccount to its parent (Account) class.

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

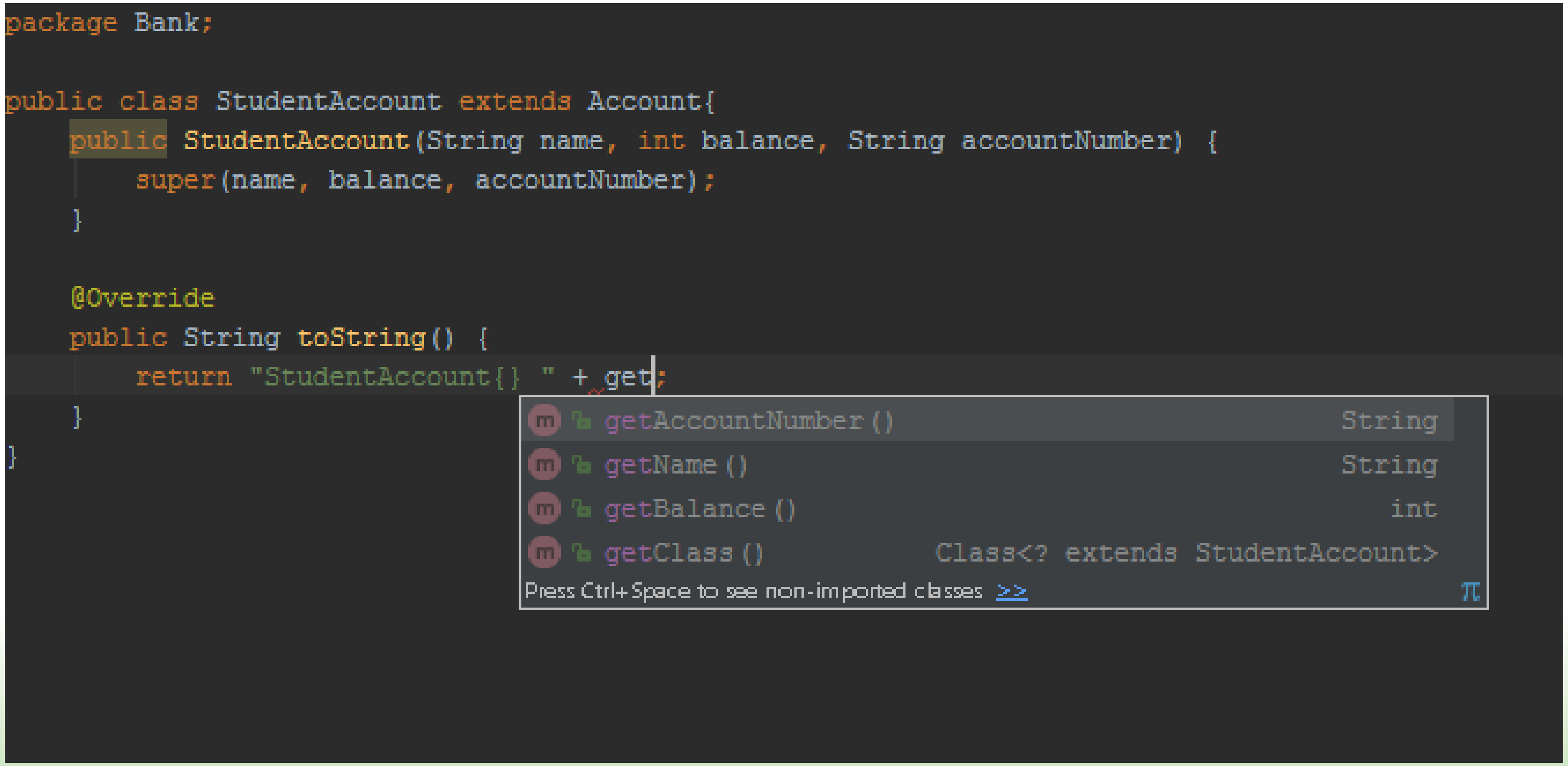
Add a toString() method to StudentAccount

```
@Override
public String toString() {
    return "StudentAccount{}";
}
```

Because we have not added any attributes to StudentAccount, there are no attributes that can be printed in the toString() method, just the basic String to be outputted.

Inheritance in Action

But, because we are extending Account, we have access to the getters and setters of the super class because they are public methods.
You can see in the image below that typing '.get' suggests methods available in the super class (Account) but not in the StudentAccount.



```
package Bank;

public class StudentAccount extends Account{
    public StudentAccount(String name, int balance, String accountNumber) {
        super(name, balance, accountNumber);
    }

    @Override
    public String toString() {
        return "StudentAccount{} " + get;
    }
}
```

```
getAccountNumber()                              String
getName()                                       String
getBalance()                                       int
getClass()          Class<? extends StudentAccount>
Press Ctrl+Space to see non-imported classes >>
```

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

Add the getter methods declared in the Account class, into the toString() method in StudentAccount to get the values, your class should now look like this.

```
package Bank;

public class StudentAccount extends Account{
    public StudentAccount(String name, int balance, String accountNumber) {
        super(name, balance, accountNumber);
    }

    @Override
    public String toString() {
        return  "Name " + getName() +
                ", Account Number " + getAccountNumber() +
                ", Balance " + getBalance();
    }
}
```

Attributes in Child Class

We can now add attributes and getters and setter methods into the child (StudentAccount) class that are unique to that class.

These attributes and methods can be used in the same way as we have done so far.

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

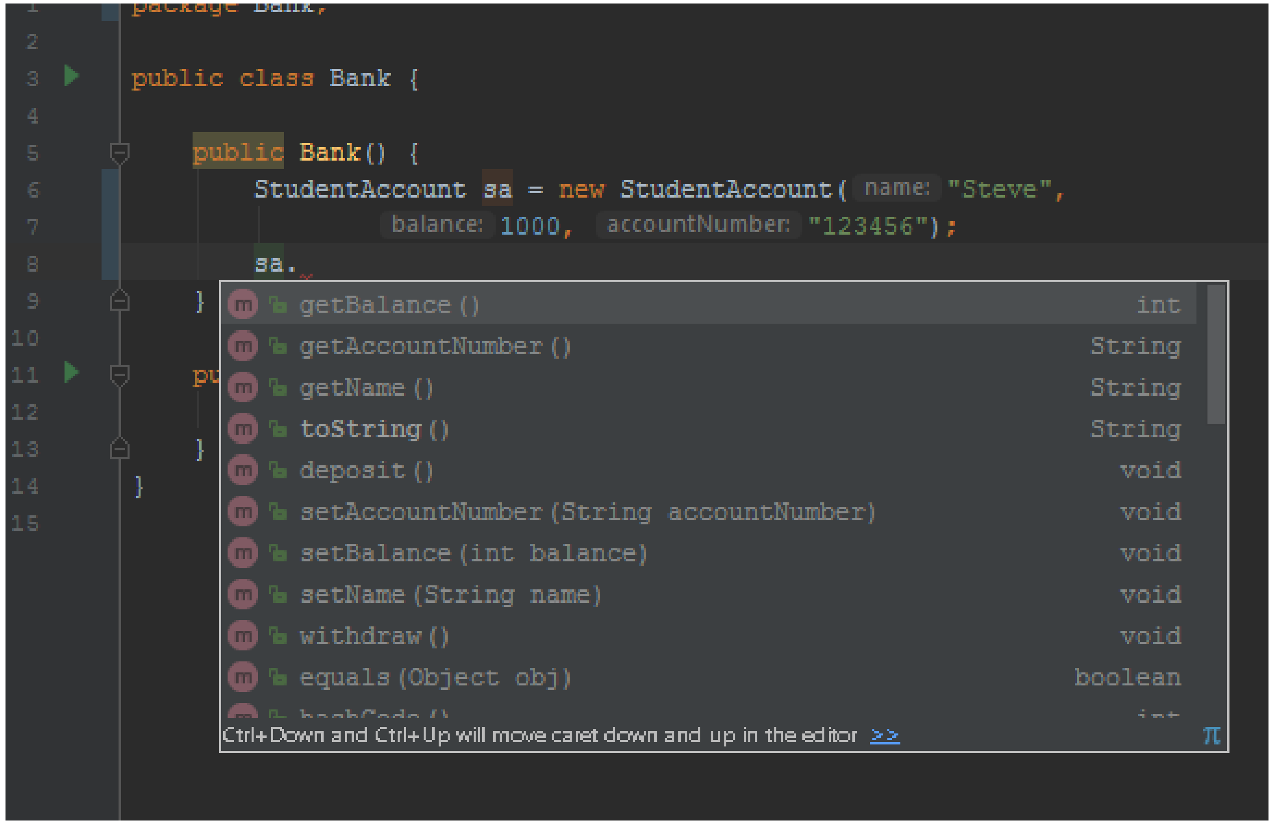
- Add a Bank class to the package
- Add a constructor to this class
- Add the main method to this class
- Add the code to create an object of the class in the main method

```
package Bank;

public class Bank {

    public Bank() {

    }

    public static void main(String[] args) {
        new Bank();
    }
}
```

Your Bank class should now look like this

Inheritance in Action

- Add the code to create an object of StudentAccount in the constructor



```
package Bank;

public class Bank {

    public Bank() {
        StudentAccount sa = new StudentAccount( name: "Steve",
                balance: 1000,  accountNumber: "123456");
        sa.
    }

    pu
    }
}
```

```
getBalance()                               int
getAccountNumber()                         String
getName()                                  String
toString()                                 String
deposit()                                  void
setAccountNumber(String accountNumber)     void
setBalance(int balance)                    void
setName(String name)                       void
withdraw()                                 void
equals(Object obj)                         boolean
hashCode()                                 int
Ctrl+Down and Ctrl+Up will move caret down and up in the editor >>  π
```

By typing the name of the object we have just created (sa) followed by . We now have access to the getters and setters of the super class (Account), we can use these methods to print out values of our object of StudentAccount (sa).

Inheritance in Action

University of HUDDERSFIELD
Inspiring tomorrow's professionals

The complete Bank code.

```java
package Bank;

public class Bank {

    public Bank() {
        StudentAccount sa = new StudentAccount("Steve",
                1000, "123456");
        System.out.println(sa.toString());
    }

    public static void main(String[] args) {
        new Bank();
    }
}
```

When we run the programme, the output windows will print the StudentAccount toString() method to screen.

```
Name Steve, Account Number 123456, Balance 1000
```

Finally

University of HUDDERSFIELD
Inspiring tomorrow's professionals

1. Download and run the code sample from Brightspace?
2. Add another type of account that follows the pattern of StudentAccount with unique attributes.
3. Continue working of outstanding tutorial work and ask me questions?

Inheritance is a crucial and integral part of any programming language, put some effort into learning how to use it. You will benefit from the effort!