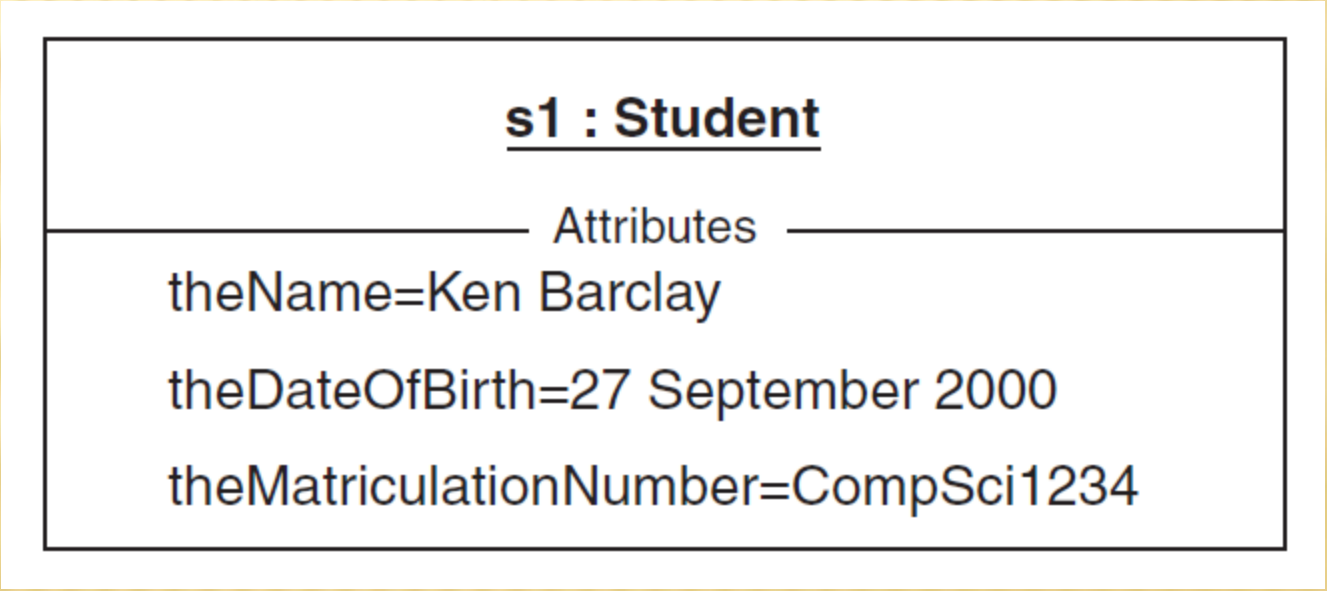
02 – Objects and Classes

OBJECT-ORIENTED PROGRAMMING

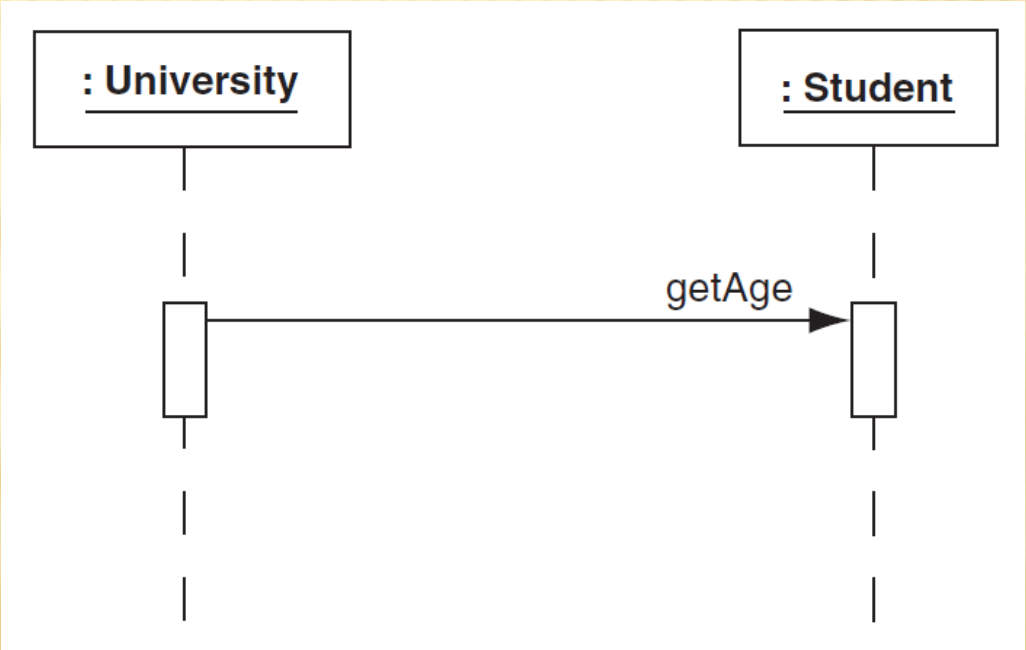
OBJECTS: COMBINED SERVICES AND DATA

- characterized by:
  - *state*: the information an object has about itself
  - *behavior*: describes the actions (*operations*) the object is prepared to engage in.



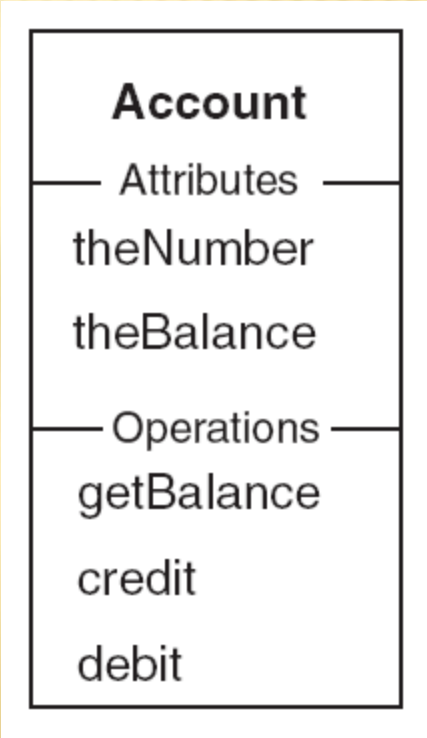
MESSAGE PASSING

- One object interacts with another:
  - perform one of its advertised *operations*
  - send a *message* to another



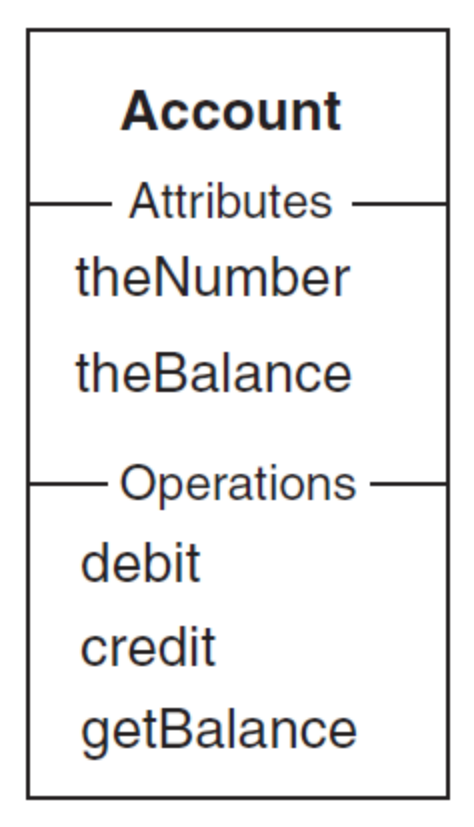
CLASSES: SETS OF SIMILAR OBJECTS

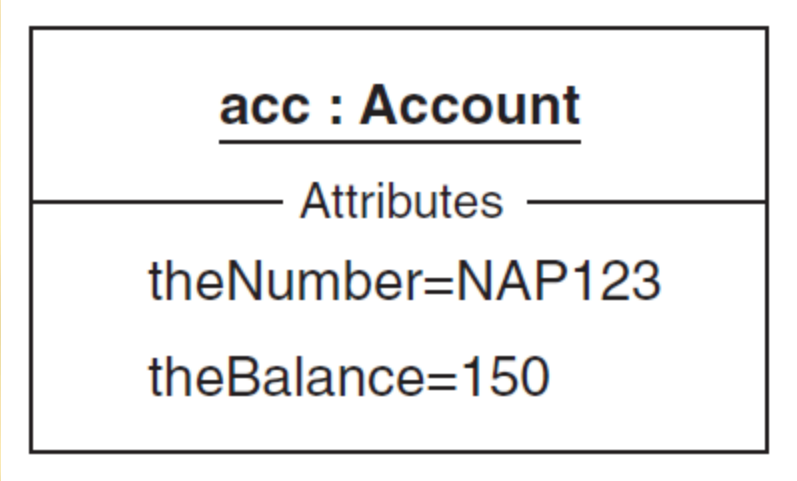
- Objects of a class:
  - same attributes and behaviors
- A blueprint (template): describe the abstraction



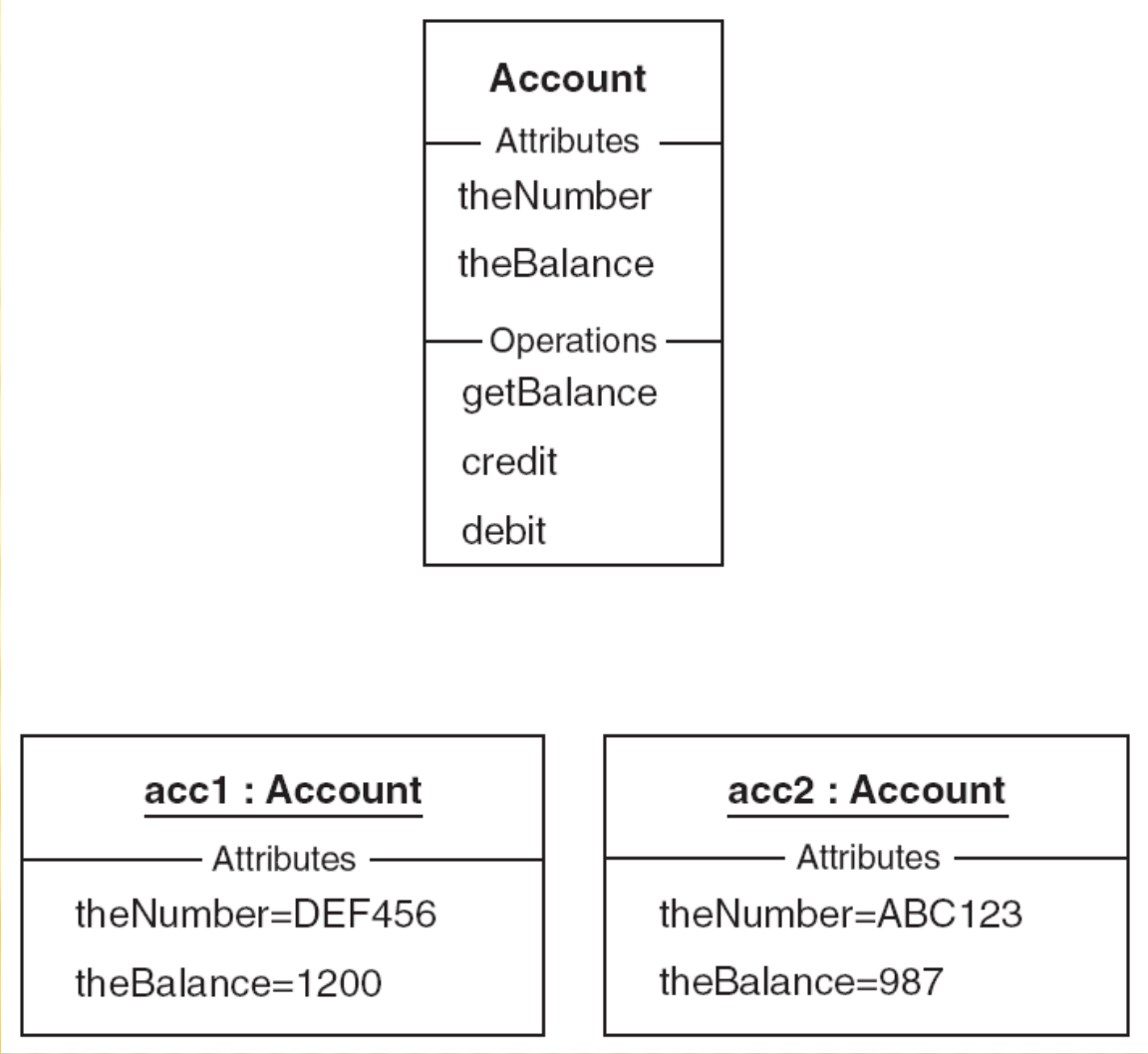
OBJECT INSTANCE

- A particular example of an object from some named class





THE ACCOUNT CLASS AND TWO INSTANCES



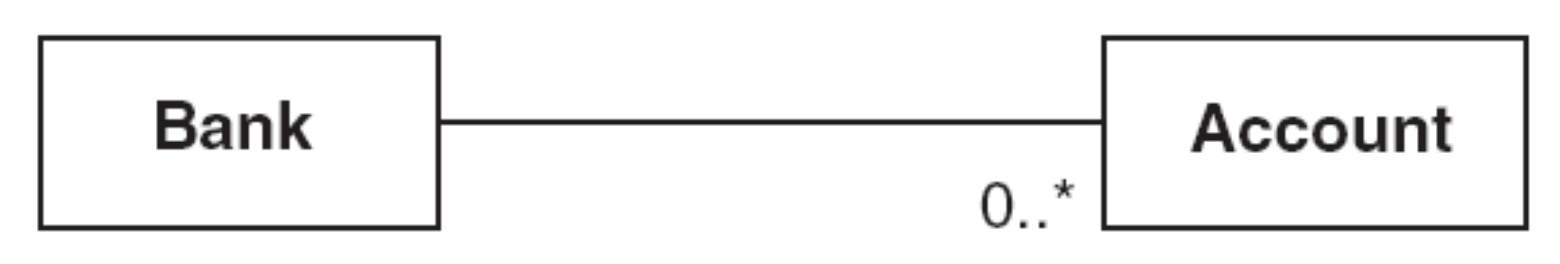
DEFINING A CLASS IN C++

```
class Account
{
  string theNumber;
  int theBalance;
};
```

DEFINING A CLASS IN JAVA

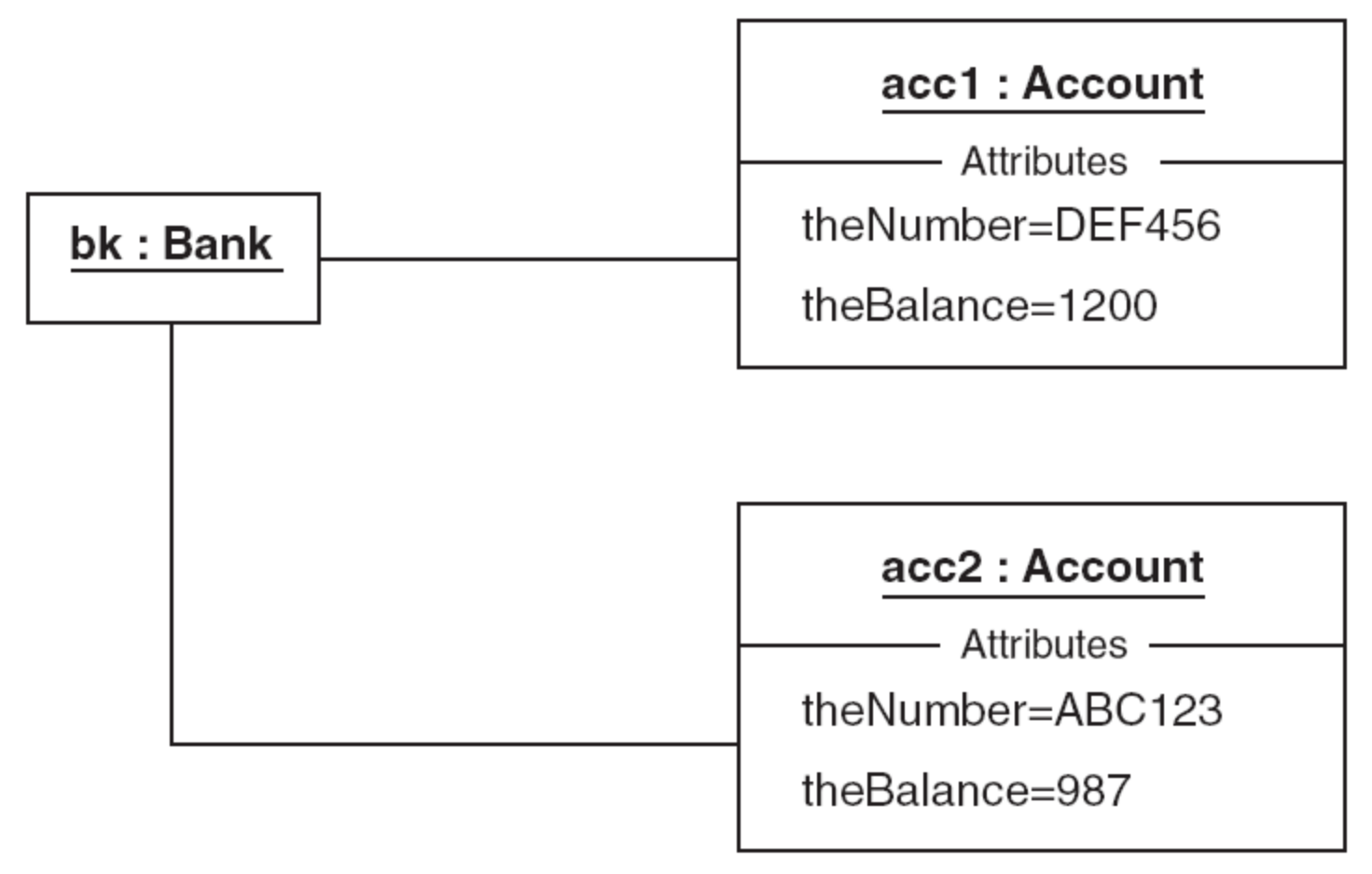
```
class Account
{
  private String theNumber;
  private int theBalance;
}
```

THE RELATIONSHIPS BETWEEN CLASSES

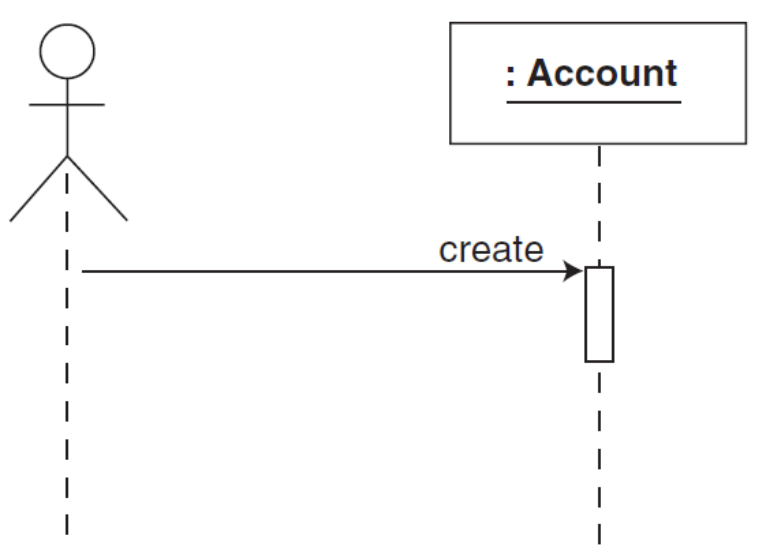


- 0..*: one Bank object can be related to none (0) or more (*) Account objects.

OBJECTS AND RELATIONSHIPS



CONSTRUCTING OBJECTS



C++	Java
Class definition: class Account { ….. };	Class definition: class Account { ….. }
Object construction: (1) Account acc(….);	
Object construction: (2) Account * p = new Account (….);	Object construction: Account acc = new Account (….);

CONSTRUCTORS

- An object is constructed by instantiating a class with the help of a class *constructor*
- set aside a part of the memory for the object
- set the various members of the object according to the arguments supplied to the constructor

CONSTRUCTORS IN C++

```
class Account
{
  string theNumber;
  int theBalance;
public:
  Account(string number, int balance) {
     theNumber = number;
     theBalance = balance;
  }
};
```

CONSTRUCTORS IN JAVA

```
class Account
{
  private String theNumber;
  private int theBalance;
  public Account(String number, int
  balance) {
     theNumber = number;
     theBalance = balance;
  }
}
```

ENCAPSULATION

- Each object instance forms a self-contained entity
- Everything an object knows is expressed in terms of its attributes
- everything it can perform is expressed by its list of operations
- objects are described as *highly cohesive*

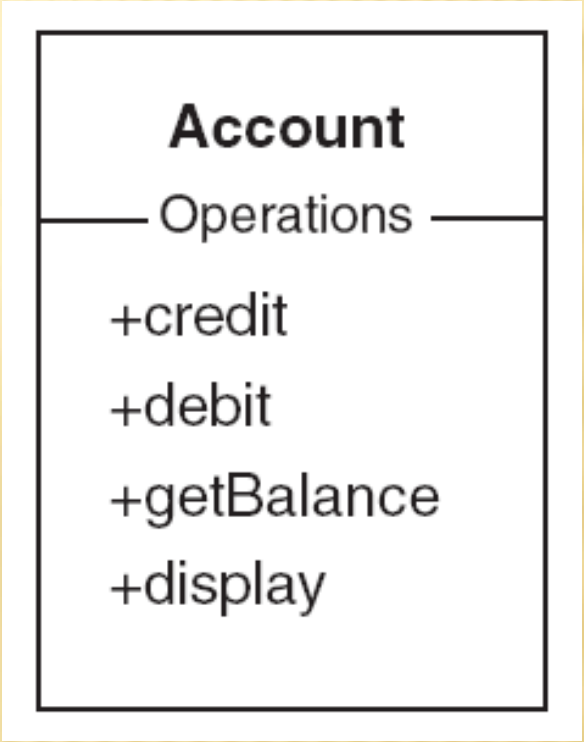
INFORMATION HIDING

```
class Account
{
   private String theNumber;
   private int theBalance;
   public int getBalance() {
      return theBalance;
   }
}
```

- object behaviour is defined by its operations and not by its private representation

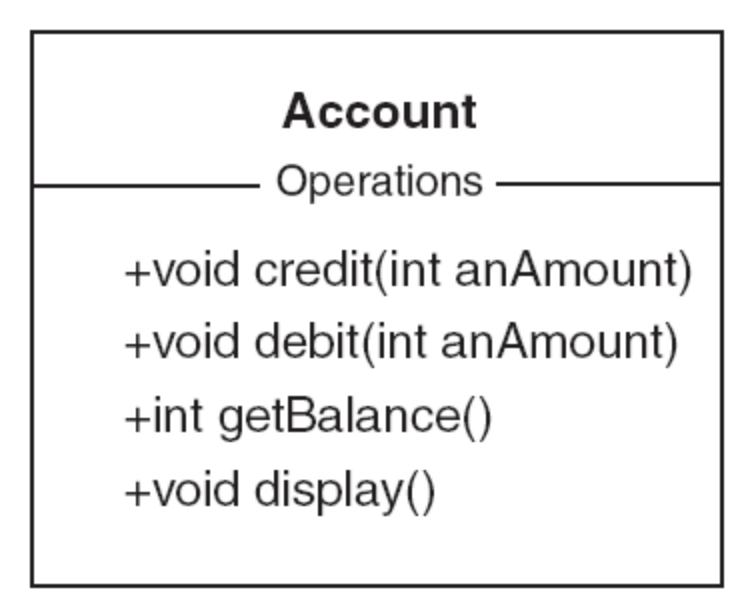
OPERATIONS

- *enquiry operations*: only give information about an object
  - + getBalance()
- *transformer operations*: changes one or more of the object instance attribute values
  - + debit()
  - + credit()



OPERATION SIGNATURE

- formal parameters
- return value



OPERATIONS IN C++

```
class Account
{
  // Operations
public:
  void credit(int anAmount) { …}
  void debit(int anAmount) { …}
  int getBalance() { …}
  void display() { …}
};
```

OPERATIONS IN JAVA

```
class Account
{
  // Operations
  public void credit(int anAmount) { …}
  public void debit(int anAmount) { …}
  public int getBalance() { …}
  public void display() { …}
}
```

THE FINAL ACCOUNT CLASS DIAGRAM