Creating a Class - BankAccount

Introduction to OO Programming

Steps in Designing a Class

1. Identify the object / objects that are required by the program
2. For each object
   a. Identify the properties of that object (instance variables)
   b. Identify the behaviours of that object (methods)
3. Create a UML class diagram
4. Implement the class (write the code for the class)
5. Write a tester containing the `main` method, where objects will be instantiated and tested.

Step 1

1. **Identify the object / objects that are required by the program**

- Normally involves identifying distinct, identifiable entities
- This may be specified in the problem statement / practical question
- In this example, we will be creating a class called BankAccount
- This will allow us to create Objects of type BankAccount, which store data and can perform operations

Step 2

2. **For each object**
   a. **Identify the properties of that object (instance variables)**
   b. **Identify the behaviours of that object (methods)**

Each BankAccount will have its own properties (instance variables) and behaviour (methods)

- Properties of a BankAccount (Instance variables)
  – balance
- Behaviour of bank account (Methods)
  – deposit money
  – withdraw money
  – get balance

Methods

- Methods of `BankAccount` class:

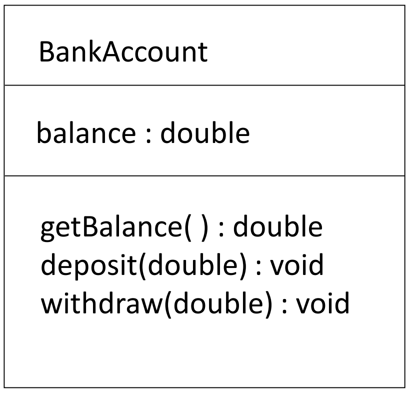
```
deposit()
withdraw()
getBalance()
```

- We need to figure out input parameter types and return types:

```
void deposit(double)
void withdraw(double)
double getBalance()
```

Step 3

3. Create a UML class diagram



Step 4

4. Implement the class (write the code for the class)

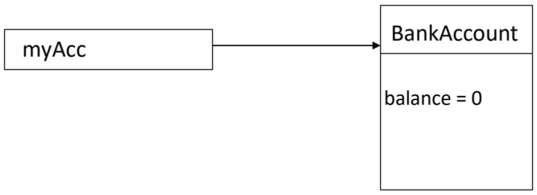
```
public class BankAccount
{

   //instance variables
   //methods

}
```

Instance Variables

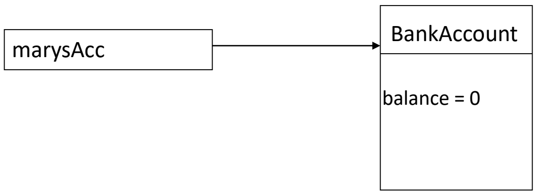
- An *instance variable* declaration consists of the following parts:
  - access specifier
    - normally `private`
  - type of variable
    - such as `double` or `int` or `String`
  - name of variable
    - such as `balance`
- Each object of a class has its own set of instance fields/variables
- You should declare all instance fields as private

Instance Variables

BankAccount myAcc = new BankAccount();



BankAccount marysAcc = new BankAccount();



Accessing Instance Fields

- Only the methods of the `BankAccount` class can access the `private` instance field `balance`

```
public void deposit(double amount)
{
    balance = balance + amount;

}
```

Implementing Methods

- Some methods do not return a value

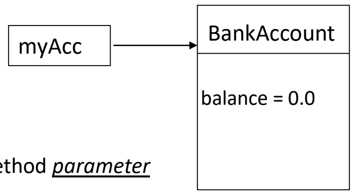
```
public void withdraw(double amount)
{
        balance = balance - amount;

}
```

- Some methods return a value

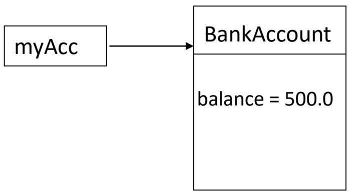
```
public double getBalance()
{
   return balance;
}
```

Method Call Example

```
myAcc.deposit(500);
```



- The value 500 is passed to the `deposit()` method *parameter variable* `amount`
- Fetch the value stored in `balance` field (0.0) of object `myAcc`
- Add the value of `amount` to `balance` and store the result in the `balance` instance field, overwriting the old value



```
public class BankAccount
{
   // instance variables
   private double balance;

   // methods
   public double getBalance()
   {
      return balance;
   }

   public void deposit(double amount)
   {
      balance = balance + amount;
   }

   public void withdraw(double amount)
   {
      balance = balance - amount;
   }
}// end class
```

Step 5

5. Write a tester containing the `main` method, where objects will be instantiated and tested.

- Write a Test class:
  - A class with a `main()` method that contains statements to test another class.
  - Must be saved in the same directory as the class
- Carry out the following steps:
  1. Construct one or more objects of the class that is being tested
  2. Invoke (or call) the methods to test them
  3. Examine the results

Testing the BankAccount class

```
File BankAccount.java

public class BankAccount
{
// instance variables


// methods


}
```

```
File BankAccountTester.java

public class BankAccountTester
{
public static void main(String[] args)
{
// declare variables
BankAccount myAcc = new BankAccount ();

myAcc.deposit(250.00);
System.out.print(myAcc.getBalance());

  }
}
```

Steps in Designing a Class (recap)

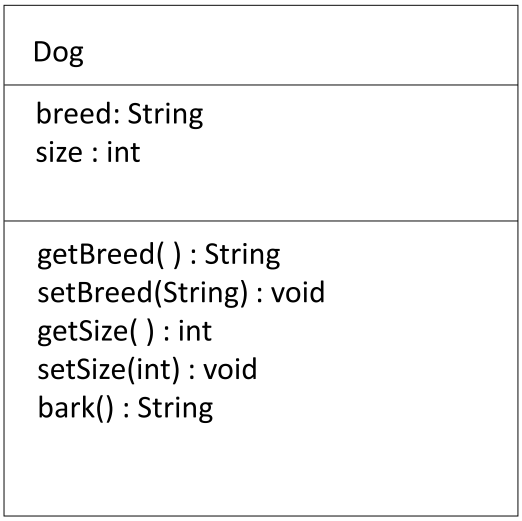
1. Identify the object / objects that are required by the program
2. For each object
   a. Identify the properties of that object (instance variables)
   b. Identify the behaviours of that object (methods)
3. Create a UML class diagram
4. Implement the class (write the code for the class)
5. Write a tester containing the `main` method, where objects will be instantiated and tested.

Another Example

Specification:

Write a class representing a Dog. Every dog will have a size in inches and a breed. Your class should include methods to set and get each instance variable. All dogs can bark.

UML Class Diagram - Dog



```java
public class Dog
{
   //instance variables
   private int size;
   private String breed;

   //methods
   public void setSize(int sizeIn)
   {
      size = sizeIn;
   }
   public int getSize()
   {
      return size;
   }
   public void setBreed(String breedIn)
   {
      breed = breedIn;
   }
   public String getBreed()
   {
      return breed;
   }
   public String bark()
   {
      return  "bow, wow";
   }
} //end Class
```

Testing the Dog class

File Dog.java

```java
public class Dog
{

// instance variables


// methods


}
```

File DogTester.java

```java
public class DogTester
{
 public static void main(String[] args)
 {
 Dog spot = new Dog();

 spot.setBreed( "Alsatian" );
 spot.setSize(23);

 System.out.print("spot says "+spot.bark());


 }
}
```