Java and C# in depth

Carlo A. Furia, Marco Piccioni, Bertrand Meyer

Java: overview by example

Bank Account

A Bank Account

- maintain a balance (in CHF) of the total amount of money
  - balance can go negative
- can open an account with an initial sum of money
- can deposit money on the account
  - deposit possible only for a nonnegative amount of money
- can withdraw money from the account
  - withdraw possible only for a nonnegative amount of money

Java implementation: BankAccount class

```
public class BankAccount {
        ...
}
```

Attribute **balance**

- maintain a balance (in CHF) of the total amount of money

```
public class BankAccount {

    // Attribute 'balance' inaccessible by clients
    private int balance;

    // Public getter for 'balance'
    public int getBalance() { return balance; }

    // Restricted setter for 'balance'
    protected void setBalance(int balance){
        this.balance = balance;
    }
        ...
}
```

Constructor: open a new account

- can open an account with an initial sum of money

```
public class BankAccount {
...
    // no-args constructor
    public BankAccount() { balance = 0;}

    // 1-arg constructor
    public BankAccount(int initialBalance) {
        if (initialBalance >= 0) {
            balance = initialBalance;
          }
        else throw new BankAccountException("…")
   }
...
}
```

Method **deposit**

- can deposit money on the account
  - deposit possible only for a nonnegative amount of money

```
public class BankAccount {
...
  // deposit 'amount'
  // don't do anything if 'amount' < 0
  public void deposit(int amount) {
      if (amount >= 0) {
          balance = balance + amount;
      }
 }
...
}
```

Method **withdraw**

- can withdraw money on the account
  - withdraw is effective only for a nonnegative amount of money.

```
public class BankAccount {
...
  // withdraw allowed 'amount'
  // access restricted only to some clients
  protected int withdraw(int amount) {
      if (amount >= 0) {
          balance = balance - amount);
          return 0;
      }
      else { return -1; }
 }
...
}
```

Premium Bank Account

A special Bank Account:

- basic functionalities as in a regular Bank Account
- has a minimum balance and a fixed fee
- if the balance goes below the minimum balance, the fee is automatically deducted from the balance
  - example:
    - minimum balance = 200, fee = 15
    - if a withdrawal brings the balance down to 150, an additional 15 is deducted, so the final balance after the deposit is 135

Java implementation:

PremiumBankAccount class inheriting from BankAccount

```
public class PremiumBankAccount
        extends BankAccount {
        ...
}
```

New attributes

- has a minimum balance and a fee

```
public class PremiumBankAccount extends BankAccount {

    public final int minimumBalance = 200;

    public final int lowBalanceFee = 15;

...
}
```

New constructor

```
public class PremiumBankAccount extends BankAccount {
...

    // constructor
    public PremiumBankAccount(int initialBalance) {
        if(initialBalance >= minimumBalance) {
            setBalance(initialBalance);}
        else{
            throw new
PremiumBankAccountException("…");          }
    }
...
}
```

Redefining withdraw

- if the balance goes below the minimum balance, the fee is automatically deducted from the balance

```
public class PremiumBankAccount extends BankAccount  {
    ...  // overrides corresponding method in BankAccount
        protected int withdraw(int amount) {
            int res = super.withdraw (amount);
            if (res == 0 && getBalance() < minimumBalance) {
                setBalance (getBalance() - lowBalanceFee);
                return 0;}
            else {if (res == -1)
                    {return -1;}
                else
                    {return 0;}
            }
```

Clients of the BankAccount Class

- A client class which runs two instances of BankAccount

```
public class BankClient {

    public static void main(String[] args) {
        BankAccount ba = new BankAccount();
        BankAccount pba = new PremiumBankAccount(250);
        System.out.println(ba.getBalance());
        System.out.println(pba.getBalance());
        ba.deposit(1800);
        pba.withdraw(100);
        System.out.println(ba.getBalance());
        System.out.println(bap.getBalance());
    }
}
```

Running a Java application

```
> javac BankAccount.java
        PremiumBankAccount.java
        BankClient.java
> java BankClient

  0
  250
  1800
  135
```