ETH Zürich
Chair of Software Engineering

Java and C# in depth

Carlo A. Furia, Marco Piccioni, Bertrand Meyer

C#: 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

C# 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;

  // Definition of setter and getter for 'balance'
  public int Balance {
      get { return balance; }
      protected set { balance = value; }
  }

...
}
```

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 virtual 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

C# implementation:

PremiumBankAccount class inheriting from BankAccount

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

New attributes

- has a minimum balance and a fee

```
public class PremiumBankAccount : BankAccount {

    public const int minimumBalance = 200;

    public const int lowBalanceFee = 15;

...
}
```

New constructor

- construction is as in the BankAccount class

```
  public class PremiumBankAccount : BankAccount {
  ...

      // constructor
      public PremiumBankAccount(int initialBalance)
            if(initialBalance >= minimumBalance) {
                  Balance = 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 : BankAccount {
    ...  // overrides corresponding method in
BankAccount
       protected override int withdraw(int amount) {
            int res = base.withdraw (amount);
            if (res == 0 && Balance < minimumBalance) {
                  Balance = Balance - 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

```
using System;
public class BankClient {

    public static void Main(String[] args) {
          BankAccount ba = new BankAccount(0);
          BankAccount bap = new PremiumBankAccount(250);
          Console.WriteLine(ba.Balance);
          Console.WriteLine(bap.Balance);
          ba1.deposit(1800);
          ba2.deposit(100);
          Console.WriteLine(ba.Balance);
          Console.WriteLine(bap.Balance);
    }
}
```

Running a C# application (under Linux)

```
> mcs bankAccount.cs
> ./bankAccount.exe

  0
  250
  1800
  135
```