Chapter 4

Defining Instantiable Classes

OBJECTIVES
After you have read and studied this chapter, you should be able to

- Define an instantiable class with multiple methods and a constructor.
- Differentiate the local and instance variables.
- Define and use value-returning methods.
- Distinguish private and public methods.
- Distinguish private and public data members.
- Describe how the arguments are passed to the parameters in method definitions.
- Use System.out for temporary output to verify the program code.

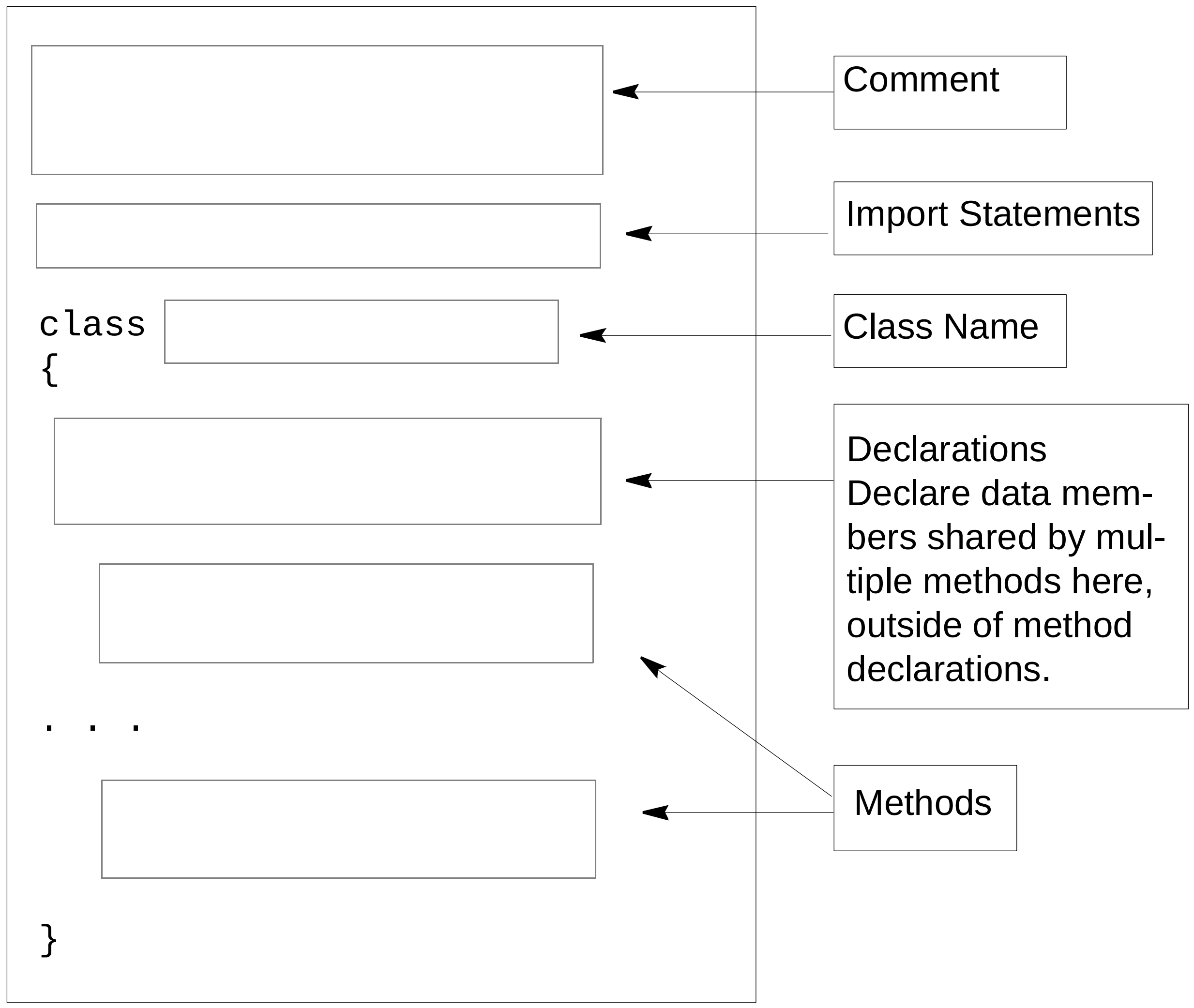
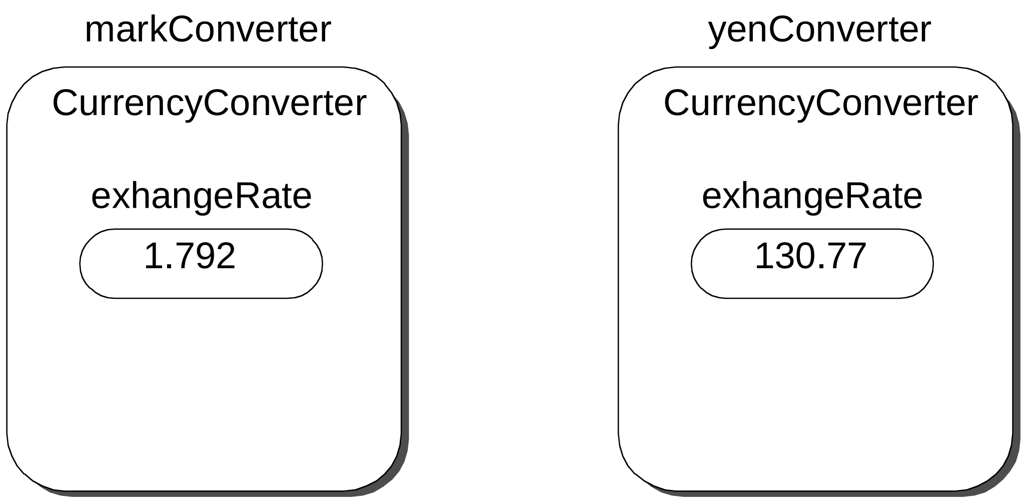

class
{
. . .
}
Comment
Import Statements
Class Name
Declarations
Declare data members shared by multiple methods here, outside of method declarations.
Methods

FIGURE 4.1 A program template for a class definition.

FIGURE 4.2 Every object of a class has its own copy of instance variables. **CurrencyConverter** objects have their own copy of **exhangeRate** instance variables.

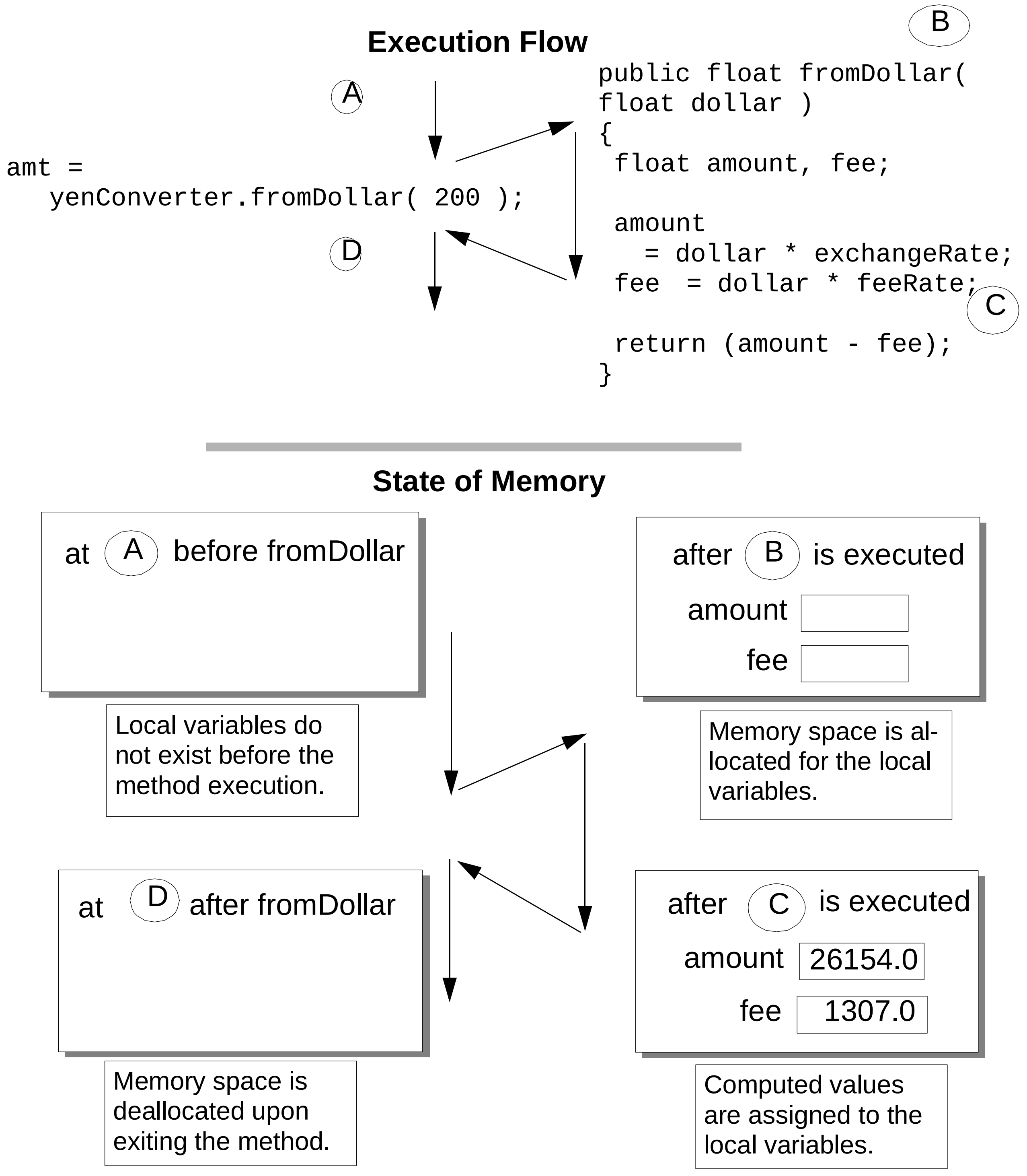
```
CurrencyConverter    markConverter, yenConverter;

markConverter = new CurrencyConverter();
markConverter.setExchangeRate(1.792f);

yenConverter = new CurrencyConverter();
yenConverter.setExchangeRate(130.77f);
```



```
/*
  Method:     setExchangeRate

  Purpose:    Sets the exchange rate to the value passed
              to this method

  Parameters:
              float rate
                  -  the exchange rate

  Returns:        None
*/
```

```
/*******************************
 Public Methods:

  floatfromDollar (float)
  floattoDollar ( float)
  void setExchangeRate ()

******************************/
```

FIGURE 4.3 How memory space for a local variable is allocated and deallocated.



```
amt =
   yenConverter.fromDollar( 200 );
```

```
public float fromDollar(
float dollar )
{
 float amount, fee;

 amount
   = dollar * exchangeRate;
 fee  = dollar * feeRate;

 return (amount - fee);
}
```

Local variables do not exist before the method execution.

Memory space is allocated for the local variables.

Computed values are assigned to the local variables.

Memory space is deallocated upon exiting the method.

FIGURE 4.4 How memory space for the parameters is allocated and deallocated.

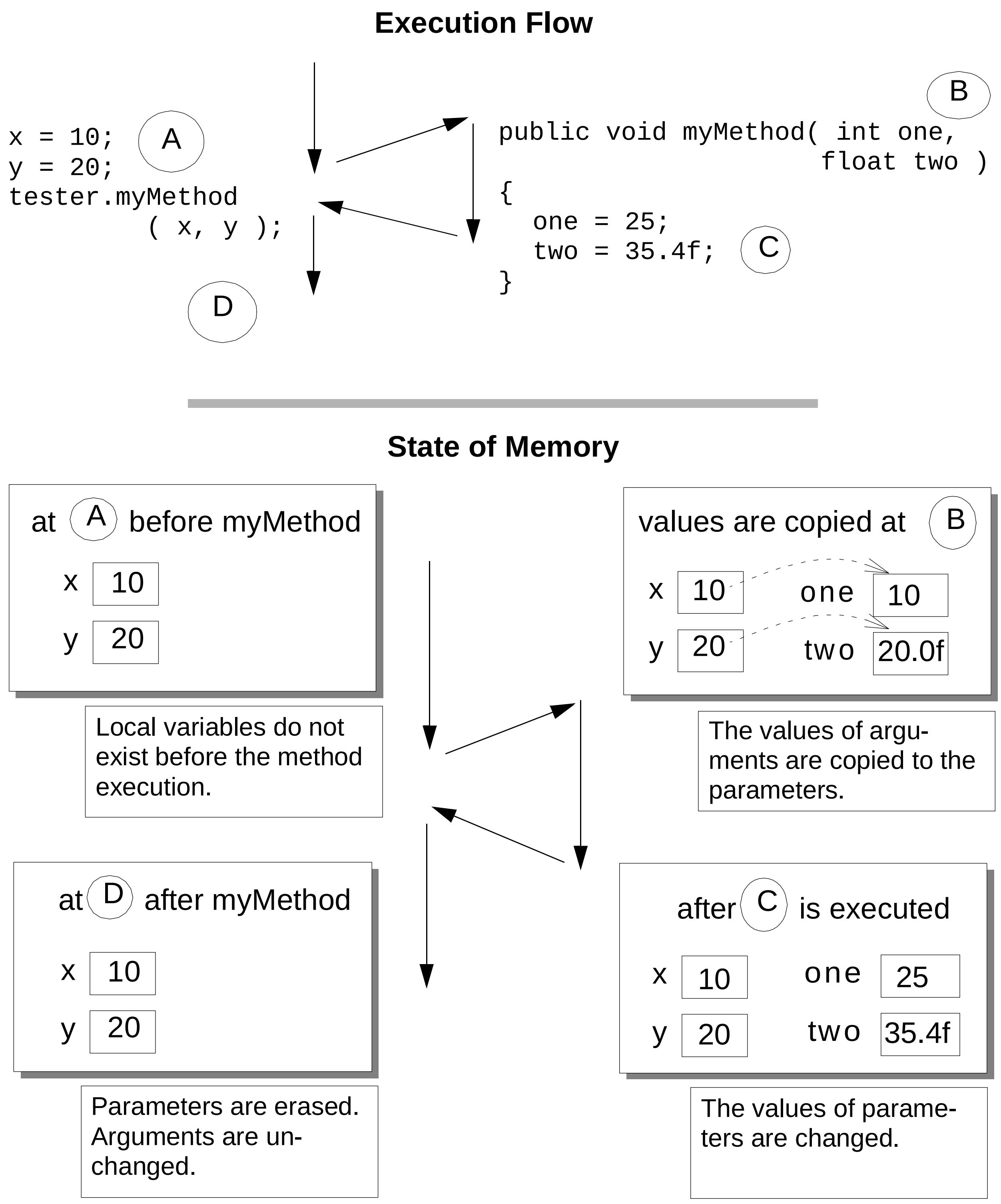


FIGURE 4.5 The object diagrams for the Chapter 3 **LoanCalculator** program and the one we are designing here. Not all methods are shown here to simplify the diagrams.

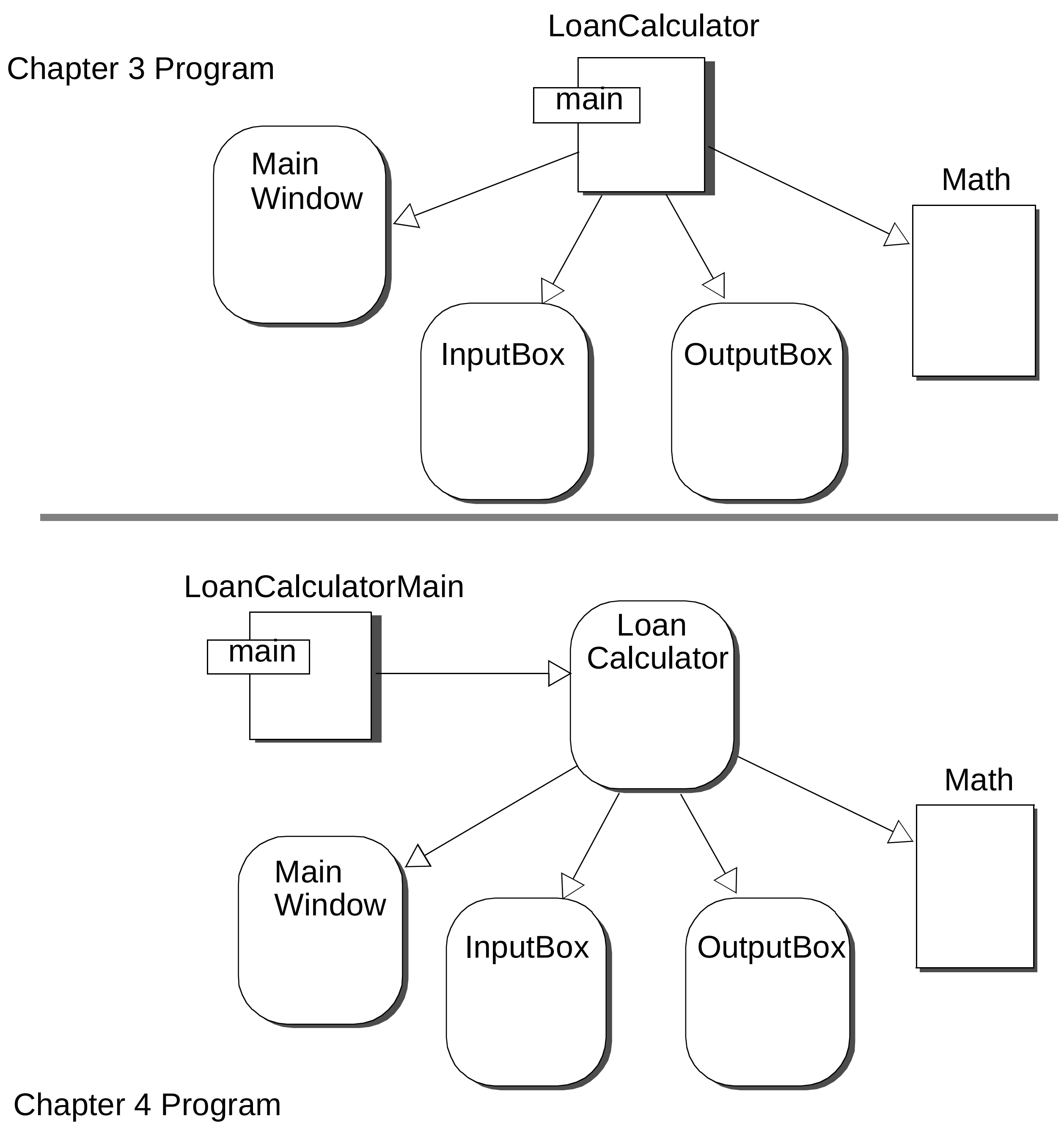


FIGURE 4.6 The object diagram for Alternative Design 1. **MainWindow**, **OutputBox**, and **InputBox** objects are not shown.

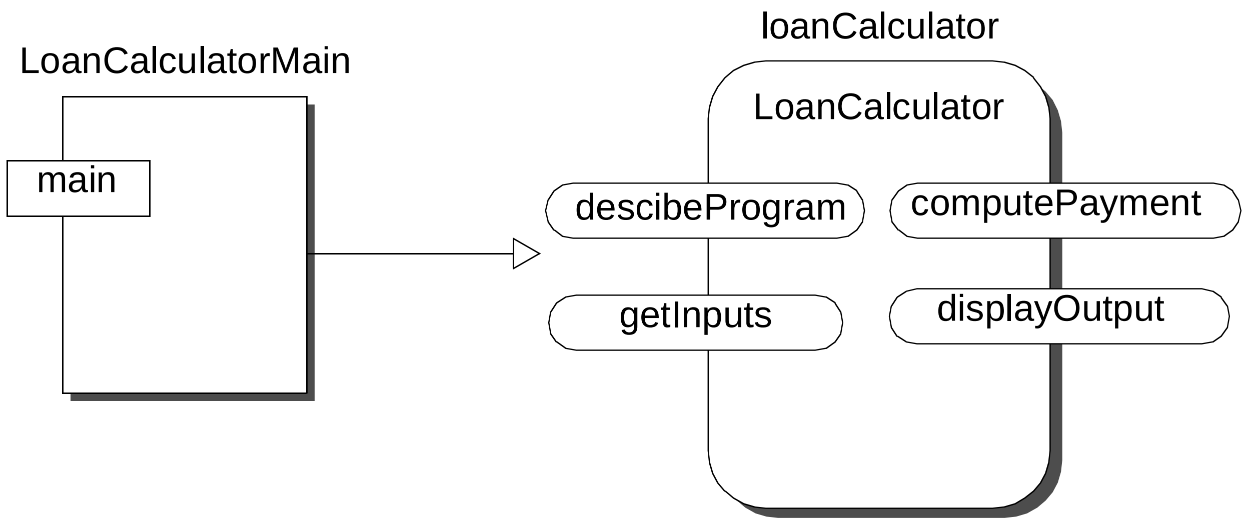


FIGURE 4.7 The object diagram for Alternative Design 2. **MainWindow**, **OutputBox**, and **InputBox** objects are not shown.

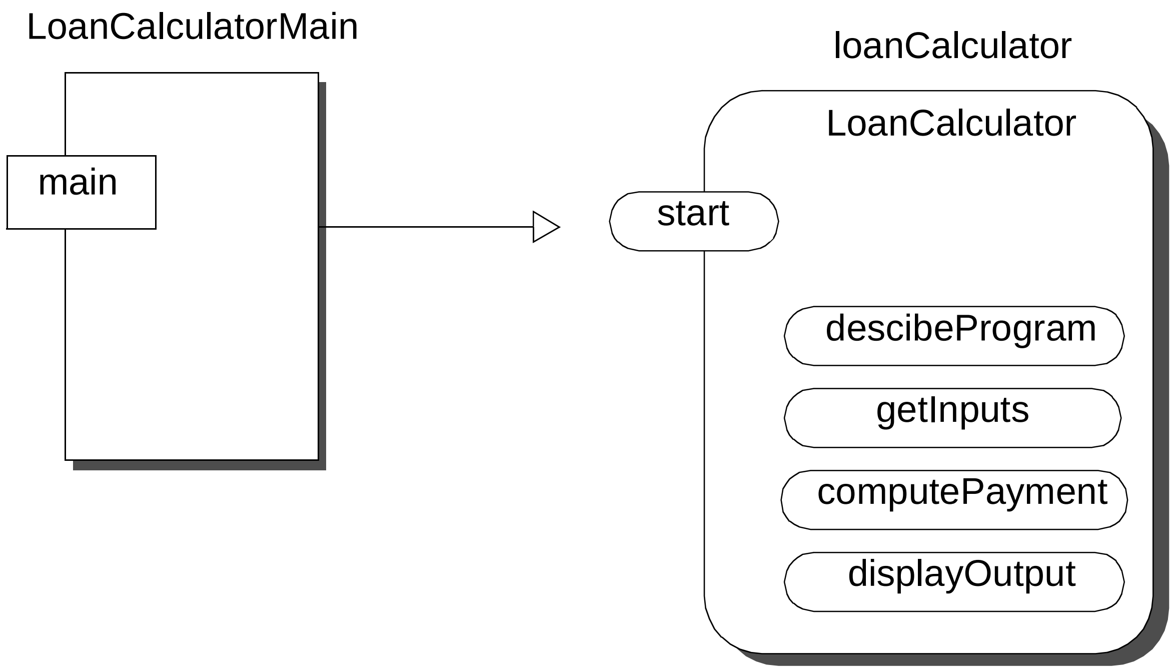
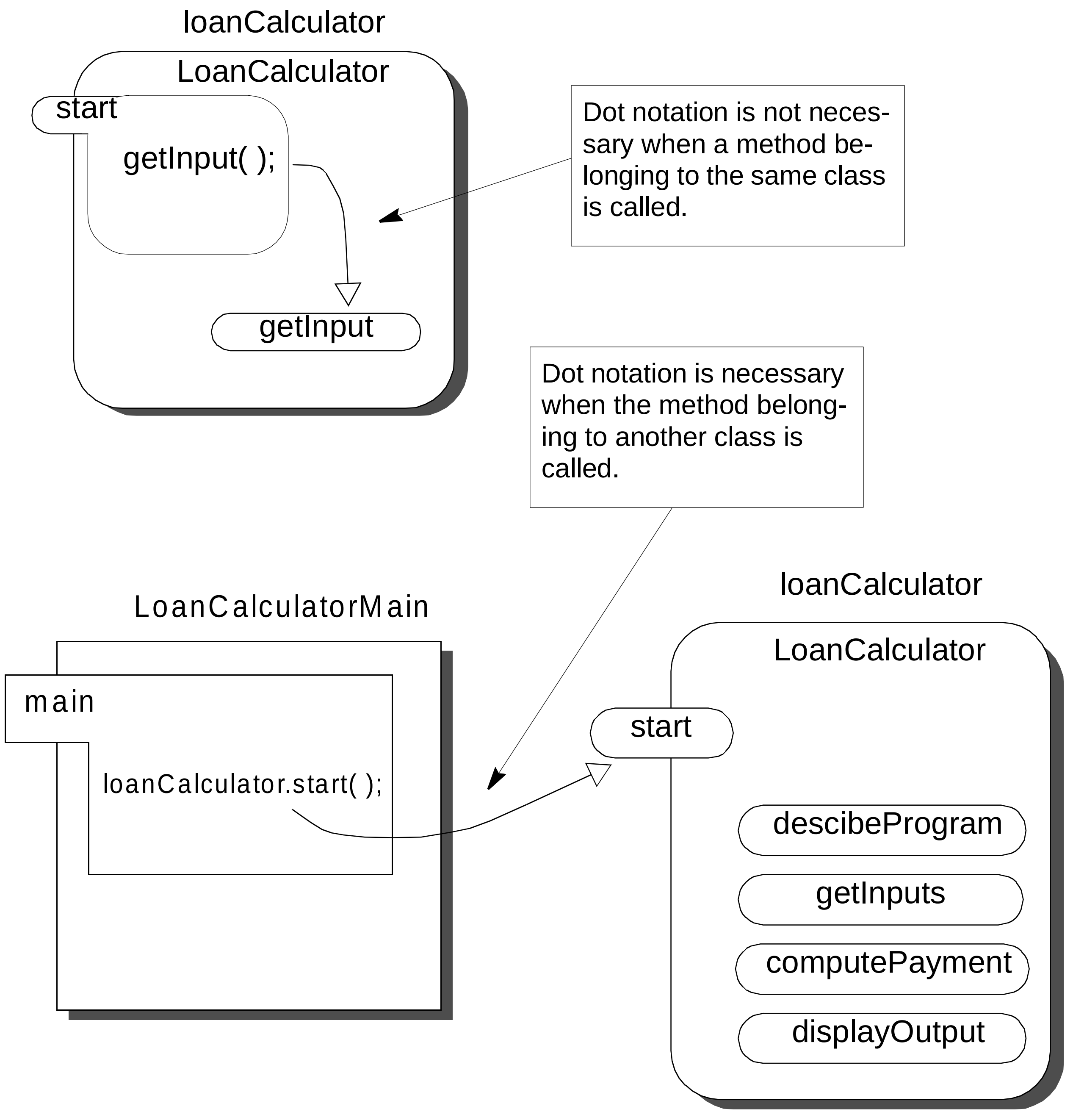


FIGURE 4.8 The difference between calling a method belonging to the same class and a method belonging to a different class.


loanCalculator
LoanCalculator
start
getInput( );
getInput
Dot notation is not necessary when a method belonging to the same class is called.
Dot notation is necessary when the method belonging to another class is called.
LoanCalculatorMain
main
loanCalculator.start( );
loanCalculator
LoanCalculator
start
descibeProgram
getInputs
computePayment
displayOutput