Chapter 5

How to structure an object-oriented application

Objectives

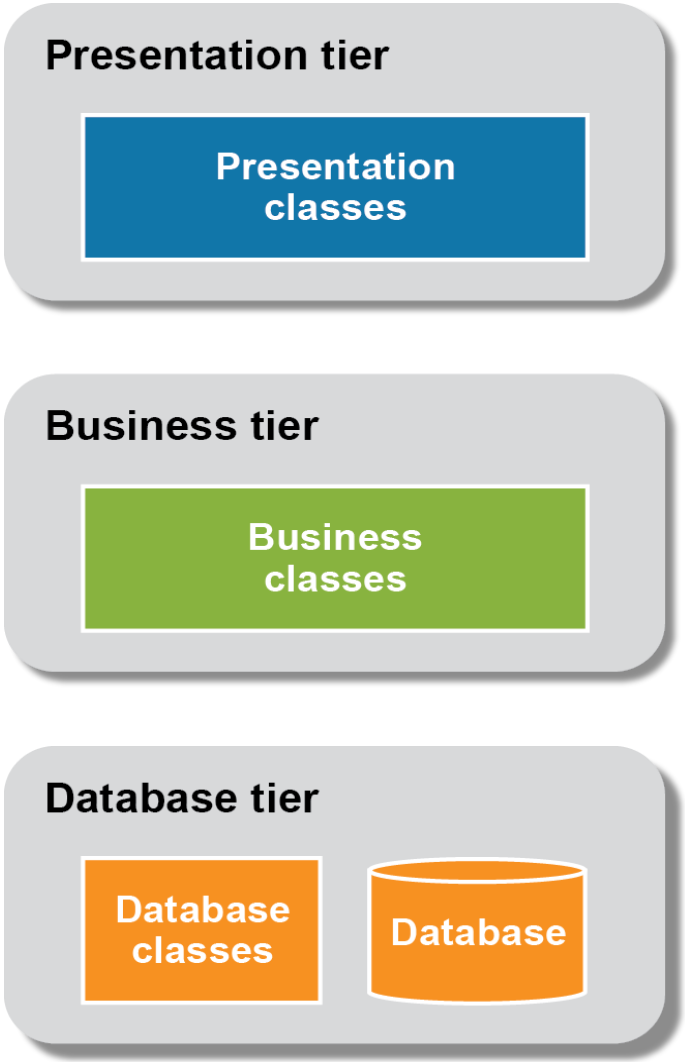
Applied

- Structure the classes in an application so they use the 3-tier architecture.
- Use NetBeans to create and work with the packages of an application.

Knowledge

- Name and describe the three tiers that make up the 3-tier architecture.
- Describe two reasons for storing classes in a package.
- Describe how the name of a package corresponds with the folders that store the package.
- Describe at least one benefit of using the 3-tier architecture.

The three-tier architecture of an application



The folders and files for an application

```
ch05_LineItem/src
    murach
        business
            LineItem.java
            Product.java
        db
            ProductDB.java
        ui
            LineItemApp.java
```

The LineItem class

```
package murach.business;

import java.text.NumberFormat;

public class LineItem {...}
```

The Product class

```
package murach.business;

import java.text.NumberFormat;

public class Product {...}
```

The ProductDB class

```
package murach.database;

import murach.business.Product;

public class ProductDB {...}
```

The LineItemApp class

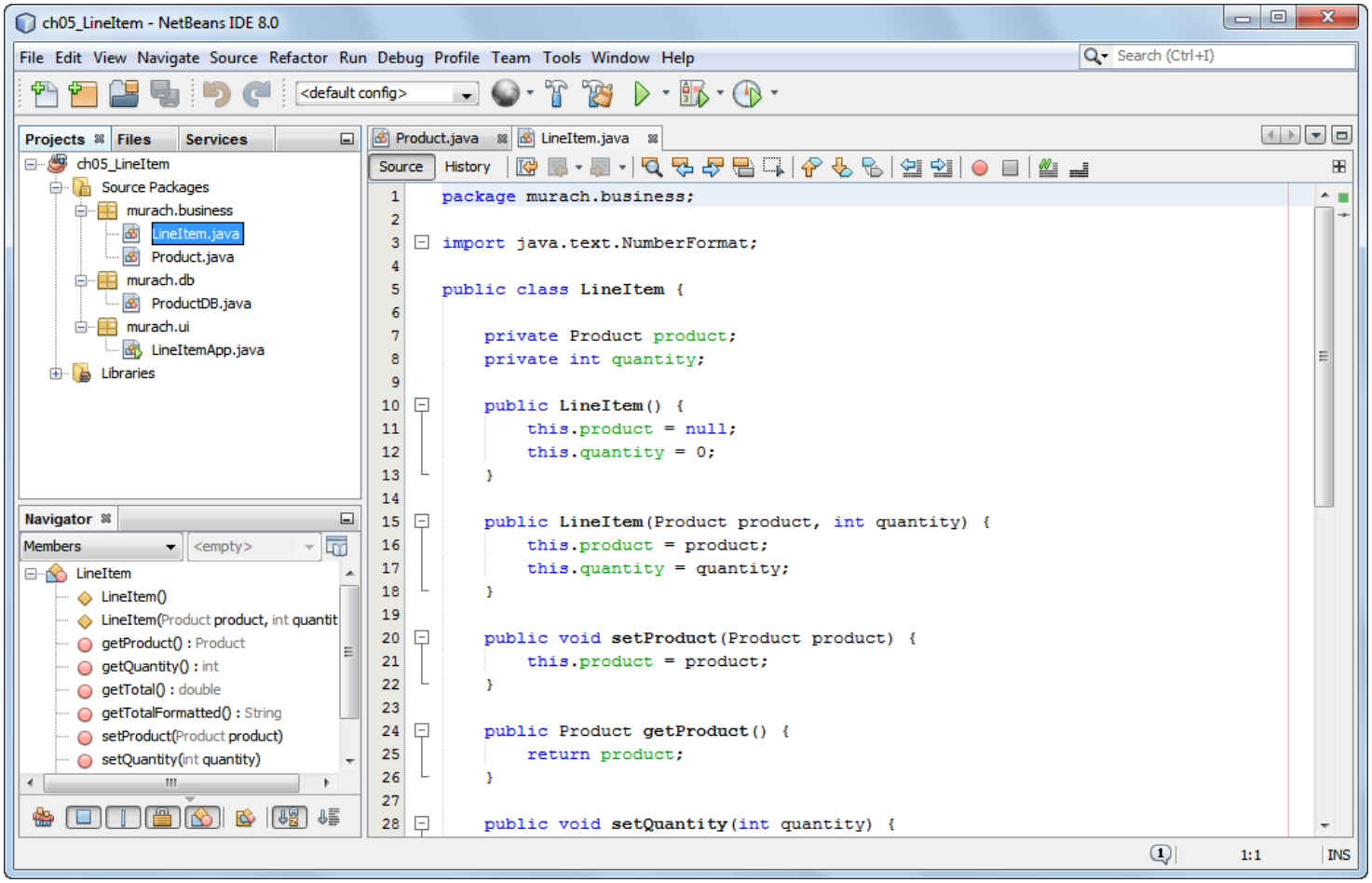
```
package murach.ui;

import java.util.Scanner;

import murach.db.ProductDB;
import murach.business.LineItem;
import murach.business.Product;

public class LineItem {...}
```

A project that contains multiple packages



The console

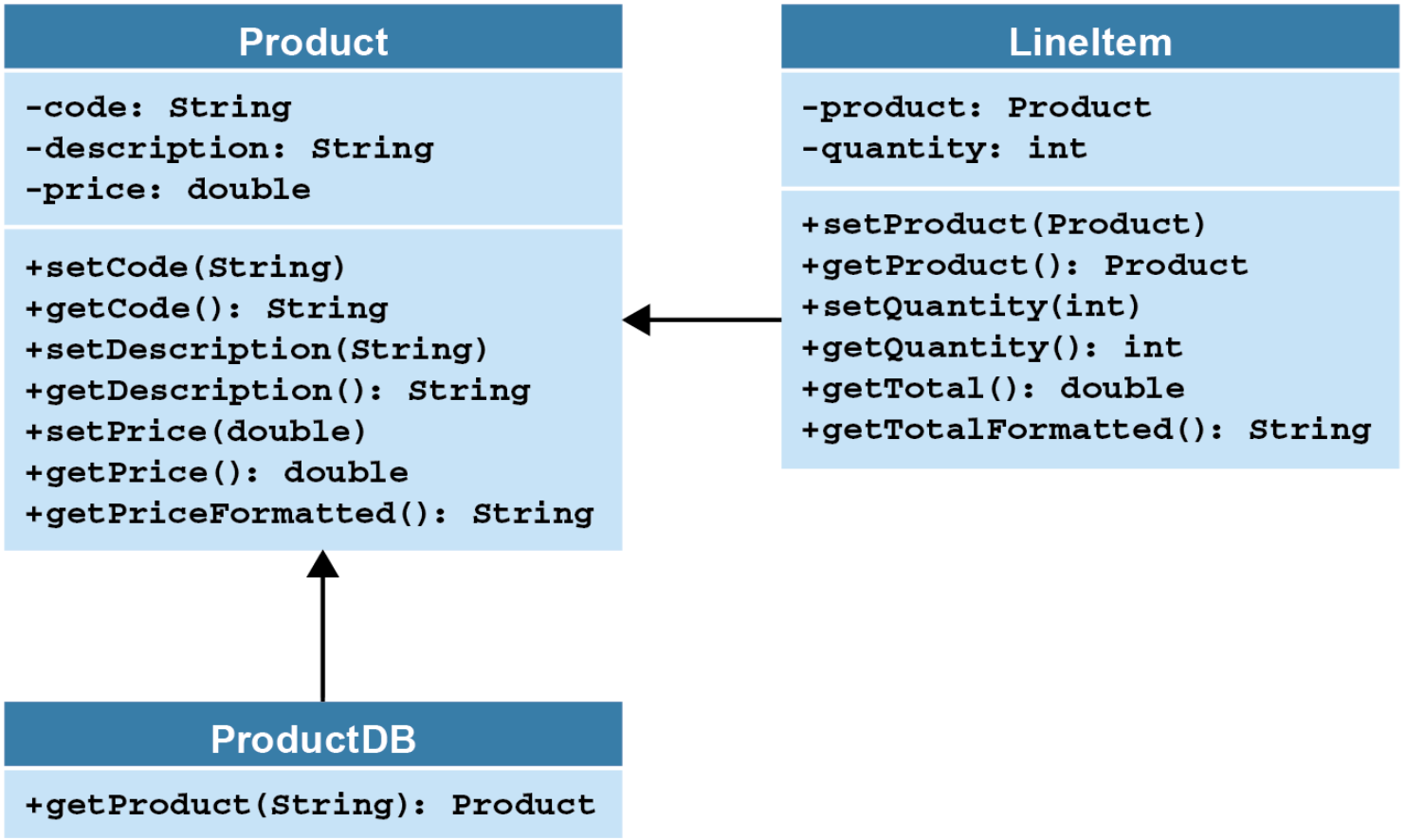
```
Welcome to the Line Item Calculator

Enter product code: java
Enter quantity:     2

LINE ITEM
Code:        java
Description: Murach's Beginning Java
Price:       $49.50
Quantity:    2
Total:       $99.00

Continue? (y/n):
```

The class diagrams



The LineItem class

```
package murach.business;

import java.text.NumberFormat;

public class LineItem {

    private Product product;
    private int quantity;

    public LineItem() {
        this.product = null;
        this.quantity = 0;
    }

    public LineItem(Product product, int quantity) {
        this.product = product;
        this.quantity = quantity;
    }
```

The LineItem class (cont.)

```
public void setProduct(Product product) {
    this.product = product;
}

public Product getProduct() {
    return product;
}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}

public int getQuantity() {
    return quantity;
}
```

The LineItem class (cont.)

```
    public double getTotal() {
        double total = quantity * product.getPrice();
        return total;
    }

    public String getTotalFormatted() {
        double total = this.getTotal();
        NumberFormat currency =
            NumberFormat.getCurrencyInstance();
        String totalFormatted = currency.format(total);
        return totalFormatted;
    }
}
```

The LineItemApp class

```
package murach.ui;

import java.util.Scanner;

import murach.db.ProductDB;
import murach.business.LineItem;
import murach.business.Product;

public class LineItemApp {

    public static void main(String args[]) {

        // display a welcome message
        System.out.println(
            "Welcome to the Line Item Calculator");
       System.out.println();
```

The LineItemApp class (cont.)

```
// create 1 or more line items
Scanner sc = new Scanner(System.in);
String choice = "y";
while (choice.equalsIgnoreCase("y")) {

    // get input from user
    System.out.print("Enter product code: ");
    String productCode = sc.nextLine();

    System.out.print("Enter quantity:     ");
    int quantity = Integer.parseInt(sc.nextLine());

    // get the Product object
    Product product =
        ProductDB.getProduct(productCode);

    // create the LineItem object
    LineItem lineItem =
        new LineItem(product, quantity);
```

The LineItemApp class (cont.)

```
            // display the output
            String message = "\nLINE ITEM\n" +
               "Code:        " + product.getCode() + "\n" +
               "Description: "
                     + product.getDescription() + "\n" +
               "Price:       "
                     + product.getPriceFormatted() + "\n" +
               "Quantity:    "
                     + lineItem.getQuantity() + "\n" +
               "Total:       "
                    + lineItem.getTotalFormatted() + "\n";
            System.out.println(message);

            // see if the user wants to continue
            System.out.print("Continue? (y/n): ");
            choice = sc.nextLine();
            System.out.println();
        }
        System.out.println("Bye!");
    }
}
```