**Self Check ch 12**

1. Suppose you sign a contract, promising that you will, for an agreed-upon price, design, implement, and test a software package exactly as it has been specified in a requirements document. What is the primary risk you and your customer are facing with this business arrangement?

**Answer:** It is unlikely that the customer did a perfect job with the requirements document. If you don’t accommodate changes, your customer may not like the outcome. If you charge for the changes, your customer may not like the cost.

1. Does Extreme Programming follow a waterfall or a spiral model?

**Answer:** An “extreme” spiral model, with lots of iterations.

1. What is the purpose of the “on-site customer” in Extreme Programming?

**Answer:** To give frequent feedback as to whether the current iteration of the product fits customer needs.

1. Suppose the invoice is to be saved to a file. Name a likely collaborator.

**Answer:** PrintStream

1. Looking at the invoice in Figure 4, what is a likely responsibility of the Customer class?

**Answer:** To produce the shipping address of the customer.

1. What do you do if a CRC card has ten responsibilities?

**Answer:** Reword the responsibilities so that they are at a higher level, or come up with more classes to handle the responsibilities

1. Consider the Bank and BankAccount classes of Chapter 7. How are they related?

 **Answer:** Through aggregation. The bank manages bank account objects

1. Consider the BankAccount and SavingsAccount objects of Chapter 10. How are they related?

**Answer:** Through inheritance.

1. Consider the BankAccountTester class of Chapter 3. Which classes does it depend on?

**Answer:** The BankAccount, System, and PrintStream classes.

1. Which class is responsible for computing the amount due? What are its collaborators for this task?

**Answer:** The Invoice class is responsible for computing the amount due. It collaborates with the LineItem class.

1. Why do the format methods return String objects instead of directly printing to System.out?

**Answer:** This design decision reduces coupling. It enables us to reuse the classes when we want to show the invoice in a dialog box or on a web page

1. Why does the Bank class in this example not store an array list of bank accounts?

**Answer:** The bank needs to store the list of customers so that customers can log in. We need to locate all bank accounts of a customer, and we chose to simply store them in the customer class. In this program, there is no further need to access bank accounts.

1. Suppose the requirements change — you need to save the current account balances to a file after every transaction and reload them when the program starts. What is the impact of this change on the design?

 **Answer:** The Bank class needs to have an additional
 responsibility: to load and save the accounts. The bank can
 carry out this responsibility because it has access to the
 customer objects and, through them, to the bank accounts.