2.3 Fields, Constructors, Methods

2.7 Does it matter whether we write

```
public class TicketMachine
```

or

```
class public TicketMachine
```

Yes, it matters. The second line produces an error and will not compile.

2.3 Fields, Constructors, Methods

2.8 Check whether or not it is possible to leave out the word public from the outer wrapper of the TicketMachine class.

No, it is not.

2.3 Fields, Constructors, Methods

2.9 List the names of the fields, constructors, and methods.

fields: price, balance, total

constructors: TicketMachine

methods: getPrice, getBalance, insertMoney, printTicket

2.3 Fields, Constructors, Methods

2.10 Do you notice any features of the constructor class that make it significantly different from the other methods of the class?

```
public TicketMachine(int ticketPrice)
```

It does not have a data type (e.g., void, int, ...)

2.3.1 Fields

2.11 What do you think is the type of each of the following fields?

private int count;

private Student representative;

private Server host;

2.3.1 Fields

2.12 What are the names of the following fields?

private boolean alive;

private Person tutor;

private Game game;

2.3.1 Fields

2.14 Is it always necessary to have a semi-colon at the end of a field declaration?

Yes.

2.3.1 Fields

2.15 Write in full the declaration for a field of type int whose name is status.

private <data type> <name of field>;

private int status;

2.3.2 Constructors

2.16 To what class does the following constructor belong?

```
public Student(String name)
```

The class name and constructor name are always the same.

This constructor must belong to the class Student.

2.3.2 Constructors

2.17 How many parameters does the following constructor have and what are their types?

```
public Book(String title, double price)
```

There are two parameters. Their types are String and double.

2.3.2 Constructors

2.18 Can you guess what types some of Book class' fields might be? Can you assume anything about the names of the fields.

```
public Book(String title, double price)
```

Since a constructor *initializes* fields, Book must have at least two fields – one of type String and one of type double.

The names of the fields are probably similar to the parameters, but not the same. For example, bookTitle and bookPrice.

Accessor Methods

2.21 Compare the getBalance and getPrice methods. What are the differences between them?

Each methods returns the value stored in a different field (balance or price). Both are integers.

Accessor Methods

2.22 If a call to getPrice can be characterized as "What do tickets cost?", how would you characterize a call to getBalance?

"How much money is left for spending on tickets?"

Accessor Methods

2.23 If the name of getBalance is changed to getAmount, does the return statement in the both of the method need to be changed, too?

No, the return statement does not care about the name of the method.

Accessor Methods

2.24 Define an accessor method, getTotal, that returns the value of the total field.

```
public int getTotal()
{
    return total;
}
```

Accessor Methods

2.26 Compare the method signatures of getPrice and printTicket in Code 2.1. Apart from their names, what is the main difference between them?

getPrice has **int** in its signature, while printTicket has **void**.

getPrice is an *accessor method*, and expects an *int* value (price) in response to its query.

printTicket does not expect a return value (void).

Accessor Methods

2.27 Do the insertMoney and printTicket methods have return statements? Why do you think this might be? Do you notice anything about their headers that might suggest why they do not require return statements?

No return statements. They do not ask a question. Their headers have **void** in them.

Mutator Methods

2.29 How can we tell from just its header that setPrice is a method and not a constructor?

public void setPrice(int ticketCost)

It specifies a data type (or in this case, a lack of data type). A constructor does not do this.

Mutator Methods

2.30 Complete the body of the setPrice method so it assigns the value of its parameter to the price field.

```
public void setPrice(int ticketCost)
{
    price = ticketCost;
}
```

Mutator Methods

2.31 Complete the body of the following method, whose purpose is to add the value of its parameter to a field named score.

```
public void increase(int points)
{
    score = score + points;
}
```

Mutator Methods

2.32 Can you complete the following method, whose purpose is to subtract the value of its parameter from a field named price?

```
public void discount(int amount)
{
    price = price – amount;
}
```

Printing from Methods

2.33 Add a method called prompt to the TicketMachine class. This should have a void return type and take no parameters.

```
public void prompt()
{
    System.out.println(“Please insert...”);
}
```

Printing from Methods

2.34 Add a showPrice method to the TicketMachine class. This should have a void return type and take no parameters.

```
public void showPrice()
{
    System.out.println(“Price is “ + price + “ cents”);
}
```

Printing from Methods

2.35 Create two ticket machines with differently priced tickets. Do calls to their showPrice methods show the same output, or different? How do you explain this effect?

The showPrice method will use the value stored in the price field when it is called (or invoked). Since each machine was created with its own price value, the output will differ as each shows its own price.

Printing from Methods

2.36 What do you think would be printed if you altered the fourth statement of printTicket so that price has quotes around it?

The output of that line would read as:

price cents.

The quotes mean that the word price is used, rather than the field price.

Printing from Methods

2.37 What about the following version?

The output would also be wrong:

price cents.

This is the same output as the previous question, but generated in a slightly different way.

Printing from Methods

2.38 Could either of the previous two versions be used to show the price of tickets in different ticket machines? Explain your answer.

No, this is not possible (with the way we are currently doing our programming). Each ticket machine object has its own copies of the fields, and they are only available within that object.