## Answers to Exercises for Chapter 10

## Exercises 10.1

1. One example of the readers and writers a multiuser chat room. Each user gets an update of the entire conversation by reading from a common database, while each user also gets to contribute to the conversation by sending an update. Another example is a medical records system where doctors and insurance companies can be concurrently reading from or writing to a patient’s record.
2. Order (producer acts before consumer) is critical in the producer-consumer problem, but is not in the readers and writers problem. Also, there can be many readers and writers, whereas there can only be one producer and one consumer.
3. To make the **Student** class thread-safe, one could define a class called **ThreadSafeStudent**, using the decorator pattern. The new class would have the same interface as **Student**, but would encase the **Student** object within a shared cell that provides the appropriate synchronization for the accessor and mutator methods. Then one could use objects of the new class in the same manner as objects of the **Student** class, but in multithreaded reader and writer situations.
4. **import time, random**

**from threading import Condition**

**class PCSharedCell(object):**

**"""Shared data that sequences reading before writing."""**

**def \_\_init\_\_(self, data):**

**"""Can produce but not consume at startup."""**

**self.data = data**

**self.writeable = True**

**self.condition = Condition()**

**def write(self, data, readerFunction):**

**"""Second caller must wait until someone has**

**consumed the data before resetting it."""**

**self.condition.acquire()**

**while not self.writeable:**

**self.condition.wait()**

**result = writerFunction(self.data)**

**self.writeable = False**

**self.condition.notify()**

**self.condition.release()**

**return result**

**def read(self, readerFunction):**

**"""Caller must wait until someone has produced**

**the data before accessing it."""**

**self.condition.acquire()**

**while self.writeable:**

**self.condition.wait()**

**result = readerFunction(self.data)**

**self.writeable = True**

**self.condition.notify()**

**self.condition.release()**

**return result**

## Exercises 10.3

1. A port is a channel through which a server receives connections from clients. An IP address uniquely identifies a server to clients or clients to a server on a network.
2. A local host is an IP address of a server’s computer used when it is not connected to a network. This IP address can be used to develop a client/server application and test it before deploying it on a network.
3. When a server creates a thread to handle a client’s request, the server is then free to listen for other client requests for service. This means that a new client will not have to wait until another client’s service has been completed.
4. An ATM could run as a client application. At startup, the client sends the user’s name and password to the server. The server replies with a successful or unsuccessful login message. If the login is successful, the other ATM commands are enabled. When the user enters a command, a special string is sent to the server to request that service. The string includes a command and any required arguments. When the server receives such a string, it performs the transaction with the bank. The server also replies to the client with the results of the transaction. When the server starts up, it loads the bank’s data from a file, instantiates the bank, and listens for client requests. When a client connects, the server creates a client socket and starts a client handler thread with that socket and the bank object.
5. There will not be a synchronization problem for access to individual accounts, as long as they are not joint accounts. If they were, then they would need a locking mechanism to ensure that an account is accessed by only one writer at a time. Because the **Bank** object uses a thread-safe dictionary to hold the accounts, the operations on the bank (as distinct from those on an individual account) are already synchronized.
6. A sever can be defined as a subclass of **Thread** and run as a separate thread from the main application. This will allow the main application to do other things while the server thread waits for client connections. The server class can also include a **quit** method. The main application can run this method to shut the server down. The **quit** method sets a Boolean instance variable to **False**. This variable is set to **True** when the thread is instantiated and can control the loop that listens for client connections.

## Exercises 10.3

1. A port is a channel through which a server receives connections from clients. An IP address uniquely identifies a server to clients or clients to a server on a network.
2. A local host is an IP address of a server’s computer used when it is not connected to a network. This IP address can be used to develop a client/server application and test it before deploying it on a network.
3. When a server creates a thread to handle a client’s request, the server is then free to listen for other client requests for service. This means that a new client will not have to wait until another client’s service has been completed.
4. An ATM could run as a client application. At startup, the client sends the user’s name and password to the server. The server replies with a successful or unsuccessful login message. If the login is successful, the other ATM commands are enabled. When the user enters a command, a special string is sent to the server to request that service. The string includes a command and any required arguments. When the server receives such a string, it performs the transaction with the bank. The server also replies to the client with the results of the transaction. When the server starts up, it loads the bank’s data from a file, instantiates the bank, and listens for client requests. When a client connects, the server creates a client socket and starts a client handler thread with that socket and the bank object.
5. There will not be a synchronization problem for access to individual accounts, as long as they are not joint accounts. If they were, then they would need a locking mechanism to ensure that an account is accessed by only one writer at a time. Because the **Bank** object uses a thread-safe dictionary to hold the accounts, the operations on the bank (as distinct from those on an individual account) are already synchronized.
6. A sever can be defined as a subclass of **Thread** and run as a separate thread from the main application. This will allow the main application to do other things while the server thread waits for client connections. The server class can also include a **quit** method. The main application can run this method to shut the server down. The **quit** method sets a Boolean instance variable to **False**. This variable is set to **True** when the thread is instantiated and can control the loop that listens for client connections.