Problem Statement

- Problem statement:

  Write a class that manages file I/O of an AddressBook object.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Development Steps

- We will develop this program in four steps:

1. Implement the constructor and the setFile method.
2. Implement the write method.
3. Implement the read method.
4. Finalize the class.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 1 Design

- We identify the data members and define a constructor to initialize them.
- Instead of storing individual Person objects, we will deal with a AddressBook object directly using Object I/O techniques.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 1 Code

Program source file is too big to list here. From now on, we ask you to view the source files using your Java IDE.

Directory: chapter09/step1

Source Files: AddressBookStorage.java
TestAddressBookStorage.java

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 1 Test

- We include a temporary output statement inside the setFile method.
- We run the test main class and verify that the setFile method is called correctly.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 2 Design

- Design and implement the write method
- The data member filename stores the name of the object file to store the address book.
- We create an ObjectOutputStream object from the data member filename in the write method.
- The write method will propagate an IOException when one is thrown.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 2 Code

Directory: chapter09/step2

Source Files: AddressBookStorage.java
TestAddressBookWrite.java

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 2 Test

- We run the test program several times with different sizes for the address book.
- We verify that the resulting files indeed have different sizes.
- At this point, we cannot check whether the data are saved correctly or not.
  - We can do so only after finishing the code to read the data back.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 3 Design

- Design and implement the read method.
- The method returns an AddressBook object read from a file (if there's no exception)
- The method will propagate an IOException when one is thrown.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 3 Code

Directory: chapter09/step3

Source Files: AddressBookStorage.java
TestAddressBookRead.java

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 3 Test

- We will write a test program to verify that the data can be read back correctly from a file.
- To test the read operation, the file to read the data from must already exist.
- We will make this test program save the data first by using the TestAddressBookWrite class from .

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Step 4: Finalize

- We perform the critical review of the final program.
- We run the final test

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.