# Self Check Questions

1. Suppose you need to read an image file that contains color values for each pixel in the image. Will you use a Reader or an InputStream?
   1. InputStream
2. Why doesn’t the URL class provide a Reader instead?
   1. Because it is faster to read the URL as binary data rather than text.
3. Why does the read method of the InputStream class return an int and not a byte?
   1. So that it can signal the end of an input with -1.
4. Decrypt the following message: Khoor/#Zruog$.
   1. Hello World!
5. Can you use the sample program from this section to encrypt a binary file, for example, an image file?
   1. Yes
6. Why doesn’t System.out support random access?
   1. Because you can’t edit what was already printed out.
7. What is the advantage of the binary format for storing numbers? What is the disadvantage?
   1. Binary storage is faster, but it is less readable by humans.
8. Why is it easier to save an object with an ObjectOutputStream than a RandomAccessFile?
   1. Because an ObjectOutputStream automatically saves each instance variable instead of having to save them manually.
9. What do you have to do to the Coin class so that its objects can be saved in an ObjectOutputStream?
   1. Implement Serializable.

# 

# Programming Exercises

P19.1: *Random monoalphabet cipher.* The Caesar cipher, which shifts all letters by a fixed amount, is far too easy to crack. Here is a better idea. For the key, don’t use numbers but words. Suppose the key word is FEATHER. Then first remove duplicate letters, yielding FEATHR, and append the other letters of the alphabet in reverse order. Write a program that encrypts or decrypts a file using this cipher. The keyword is specified with the -k command line option. The -d command line option specifies decryption. For example, java Encryptor -d -k FEATHER encrypt.txt output.txt decrypts a file using the keyword FEATHER. It is an error not to supply a keyword.

P19.2: *Letter frequencies.* If you encrypt a file using the cipher of Exercise P19.1, it will have all of its letters jumbled up, and will look as if there is no hope of decrypting it without knowing the keyword. Guessing the keyword seems hopeless, too. There are just too many possible keywords. However, someone who is trained in decryption will be able to break this cipher in no time at all. The average letter frequencies of English letters are well known. The most common letter is E, which occurs about 13% of the time. Write a program that reads an input file and prints the letter frequencies in that file. Such a tool will help a code breaker. If the most frequent letters in an encrypted file are H and K, then there is an excellent chance that they are the encryptions of E and T.

P19.8: Write a program that manipulates a database of product records. Records are stored in a binary file. Each record consists of these items:

* Product name: 30 characters at two bytes each = 60 bytes
* Price: one double = 8 bytes
* Quantity: one int = 4 bytes

# Project

Project 19.2: Write a toolkit that helps a cryptographer decrypt a file that was encrypted using a monoalphabetic cipher. A monoalphabetic cipher encrypts each character separately. Examples are the Caesar cipher and the cipher in Exercise P19.1. Analyze the letter frequencies as in Exercise P19.2. Use brute force to try all Caesar cipher keys, and check the output against a dictionary file. Allow the cryptographer to enter some substitutions and show the resulting text, with the unknown characters represented as ?. Try out your toolkit by decrypting files that you get from your classmates.