CHAPTER EXERCISES

Section 10.1

1. What are the two kinds of entities "bundled" in a class?
2. What kind of entity can there be any number of instances created for a given class?
3. What are the three fundamental features that object-oriented programming languages have in support of object-oriented programming?

Section 10.2

4. What else does encapsulation provide other than the ability to bundle together instance variables and methods?
5. Describe what it means for a member of a class to be private.
6. Explain the purpose of getters and setters.
7. Explain what the special identifier `self` is used for in Python.
8. Explain the use of name mangling in Python.
9. Explain when special methods `__str__` and `__repr__` are each used in Python.
10. Give an implementation of special method `__str__` for a `Range` class (representing a range of integers) that contains integer instance variables, `__start` and `__end`, so that the value of `Range` objects are displayed as follows: `'10...16'`, when output with `print`.
11. Give an implementation of special method `__lt__` for the `Range` class of exercise 10 so that `range1 < range2` evaluates to `True` if all the values in `range1` are less than all the values in `range2`, and returns `False` otherwise.

Section 10.3

12. Give the one-line class definition header for a class named `MySubclass` that is a subclass of the class `MySuperclass`.
13. Explain when a subclass can serve as a subtype.
14. For an object `obj`, show how in Python the type of the object may be determined from within a program or the Python shell.
15. Show how in the Python shell information about one of the built-in types of Python can be displayed.

Section 10.4

16. Explain the concept of polymorphism in object-oriented programming.
17. Explain the advantages of using polymorphism in program design.
18. What is meant by "duck typing" in Python?

Section 10.5

19. What does the name UML stand for?
20. What are the two types of diagrams in UML mentioned in the chapter, and what aspects of a program design does each represent?
21. Give a class diagram for the `XYCoord` class in the Let's Try It box of section 10.2.2.
22. Give a class diagram that includes the partial description of the built-in `str` type in Figure 10-15, and the `ExplodedStr` subclass in Figure 10-16.
23. Give a class diagram for the `Fraction` class developed in section 10.2. Include the `MixedFraction` subclass in the diagram.

PYTHON PROGRAMMING EXERCISES

P1. Give a UML class diagram for a library. Include as entities tangible objects (such as books), persons (such as borrowers and librarians), and status (such as whether a book is checked out or not). Use multiplicity, navigation, and role names where appropriate.

P2. Design and implement a `Money` class that stores monetary values in dollars and cents. Special method `__init__` should have the following function header,

```
def __init__(self, dollars, cents)
```

Include special method `__repr__` (`__str__`) for displaying values in dollars and cents: `$ 0.45`, $ 1.00, $ 1.25. Also include special method `__add__`, and three getter methods that each provide the monetary value in another currency. Choose any three currencies to convert to.

P3. Implement a class named `AvgList` as a subclass of the built-in list class in Python, able to compute the average of a list of numeric values. If the list contains any nonnumeric types, a `ValueError` exception should be raised.

P4. Design and implement a `FootMeasure` class that stores a linear measurement of feet and inches. Your class should have the following function header for special method `__init__`,

```
def __init__(self, feet=0, inches=0)
```

Thus, the class should be able to create a `FootMeasure` object in various ways by use of optional keyword arguments,

```
meas = FootMeasure()
meas = FootMeasure(feet=5)
meas = FootMeasure(feet=5, inches=8)
meas = FootMeasure(inches=68)
```

Implement special method `__repr__` in the class so that measurements are displayed as follows,

```
5 ft.          NOT    5 ft. 0 in.
5 ft. 8 in.    NOT    68 in.
```

When the measurement is 0, it should be displayed as, `0 ft. 0. ins.` Include special method `add()` for adding `FootMeasure` values. Also include all the special methods for implementing the relational operators.

P5. Develop an abstract class named `Temperature` that stores a single temperature. The class should have the following function header for special method `__init__`,

```
def __init__(self, temperature)
```

The abstract class should contain the following methods:

`__str__` — should return a string of the form "75 degrees Fahrenheit"

`aboveFreezing()` — returns `True` if temperature above the freezing point

`convertToFahren` — returns a new `Temperature` object converted to degrees Fahrenheit

`convertToCelsius` — returns a new `Temperature` object converted to degrees Celsius

`convertToKelvin` — returns a new `Temperature` object converted to degrees Kelvin