I

A) Write a program with method called swapPairs that accepts a String as a parameter and returns that String with each pair of adjacent letters reversed. If the String has an odd number of letters, the last letter is unchanged. For example, the call swapPairs("example") should return "xemalpe" and the call swapPairs("hello there") should return "ehll ohtree".

B) Write a class called RationalNumber that represents a fraction with an integer numerator and denominator.

A RationalNumber object should have the following methods:

public RationalNumber(int numerator, int denominator)

Constructs a new rational number to represent the ratio (numerator/denominator). The denominator cannot be 0, so throw an IllegalArgumentException if 0 is passed.

public RationalNumber()

Constructs a new rational number to represent the ratio (0/1).

public int getDenominator()

Returns this rational number's denominator value; for example, if the ratio is (3/5), returns 5.

public int getNumerator()

Returns this rational number's numerator value; for example, if the ratio is (3/5), returns 3.

public String toString()

Returns a String representation of this rational number, such as "3/5". You may wish to omit denominators of 1, returning "4" instead of "4/1".

An extra challenge would be to maintain your RationalNumber objects in reduced form, avoiding rational numbers such as 3/6 in favor of 1/2, or avoiding 2/–3 in favor of –2/3. Another possible extra feature would be methods to add and multiply two rational numbers.

C) Write an inheritance hierarchy that stores data about sports players. Create a common superclass and/or interface to store information common to any player regardless of sport, such as name, number, and salary. Then create subclasses for players of your favorite sports, such as basketball, soccer, or tennis. Place sport-specific information and behavior (such as kicking or vertical jump height) into subclasses whenever possible.

II

A) Write a program with method called repl that accepts a String and a number of repetitions as parameters and returns the String concatenated that many times. For example, the call repl("hello", 3) should return "hellohellohello". If the number of repetitions is zero or less, the method should return an empty string.

B) Write a class called Date that represents a date consisting of a year, month, and day. A Date object should have the following methods:

public Date(int year, int month, int day) -Constructs a new Date object to represent the given date.
public void addDays(int days)-Moves this Date object forward in time by the given number of days.
public void addWeeks(int weeks) -Moves this Date object forward in time by the given number of seven-day weeks.
public int daysTo(Date other)-Returns the number of days that this Date must be adjusted to make it equal to the given other Date.
public int getDay()-Returns the day value of this date; for example, for the date 2006/07/22, returns 22.
public int getMonth()-Returns the month value of this date; for example, for the date 2006/07/22, returns 7.
public int getYear()-Returns the year value of this date; for example, for the date 2006/07/22, returns 2006.
public boolean isLeapYear()

C) Declare an interface called Incrementable which represents items that store an integer that can be incremented in some way. The interface has a method called increment that increments the value and a method called getValue that returns the value. Once you have written the interface, write two classes called SequentialIncrementer and RandomIncrementer that implement the interface. The SequentialIncrementer begins its value at 0 and increases it by 1 each time it is incremented. The RandomIncrementer begins its value at a random integer and changes it to a new random integer each time it is incremented.

III

A) Write a program that prompts for two people's birthdays (month and day), along with today's month and day. The program should figure out how many days remain until each user's birthday and which birthday is sooner. Hint: It is much easier to solve this problem if you convert each date into an "absolute day" of year, from 1 through 365.

B) Write a Java class Clock for dealing with the day time represented by hours, minutes, and seconds. Your class must have the following features:

Three instance variables for the hours (range 0 - 23), minutes (range 0 - 59), and seconds (range 0 - 59).

- Three constructors:
  - default (with no parameters passed; is should initialize the represented time to 12:0:0)
  - a constructor with three parameters: hours, minutes, and seconds.
- *get*-methods getHours(), getMinutes(), getSeconds() with no parameters that return the corresponding values.
- *set*-methods setHours(), setMinutes(), setSeconds() with one parameter each that set up the corresponding instance variables.
- method tick() with no parameters that increments the time stored in a Clock object by one second.
- method addClock() accepting an object of type Clock as a parameter. The method should add the time represented by the parameter class to the time represented in the current class.

C) Write an inheritance hierarchy of three-dimensional shapes. Make a top-level shape interface that has methods for getting information such as the volume and surface area of a three-dimensional shape. Then make classes and subclasses that implement various shapes such as cubes, rectangular prisms, spheres, triangular prisms, cones, and cylinders. Place common behavior in super classes whenever possible, and use abstract classes as appropriate. Add methods to the subclasses to represent the unique behavior of each three-dimensional shape, such as a method to get a sphere's radius.

IV

A) Write a program that plays a guessing game with the user. The program should generate a random number between 1 and some maximum (such as 100), then prompt the user repeatedly to guess the number. When the user guesses incorrectly, the game should give the user a hint about whether the correct answer is higher or lower than the guess. Once the user guesses correctly, the program should print a message showing the number of guesses that the user made.

B) Write a Java class Complex for dealing with complex number. Your class must have the following features:

- Instance variables :
  - **realPart** for the real part of type double
  - **imaginaryPart** for imaginary part of type double.
- Constructor:
  - **public Complex ()**: A default constructor, it should initialize the number to 0, 0)
  - **public Complex (double realPart, double imaginaryPart)**: A constructor with parameters, it creates the complex object by setting the two fields to the passed values.
  - **public Complex multiply (Complex otherNumber)**: This method will find the product of the current complex number and the passed complex number. The methods returns a new Complex number which is the product of the two.

C) Write an inheritance hierarchy to model items at a library. Include books, magazines, journal articles, videos, and electronic media such as CDs. Include in a superclass and/or interface common information that the library must have for every item, such as a unique identification number and title. Place behavior and information that is specific to items, such as a video's runtime length or a CD's musical genre, into the subclasses.

V

A) Write a method called season that takes as parameters two integers representing a month and day and returns a String indicating the season for that month and day. Assume that the month is specified as an integer between 1and 12 (1 for January, 2 for February, and so on) and that the day of the month is a number between 1 and 31. If the date falls between 12/16 and 3/15, the method should return "winter". If the date falls between 3/16 and 6/15, the method should return "spring". If the date falls between 6/16 and 9/15, the method should return "summer". And if the date falls between 9/16 and 12/15, the method should return "fall".

B)Write a Java class Book with following features:

- Instance variables :
  - **title** for the title of book of type String.,**author** for the author's name of type String, **price** for the book price of type double.
- Constructor:
  - **public Book (String title, Author name, double price)**: A constructor with parameters, it creates the Author object by setting the the fields to the passed values.
- Instance methods:
  - **public void setTitle(String title)**: Used to set the title of book and **public String getTitle()**: This method returns the title of book.

Write a separate class **BookDemo** with a main() method creates a Book titled "Developing Java Software". Prints the Book's string representation to standard output (using System.out.println).

C)Create base class Teacher and a sub class PhysicsTeacher. Sinceclass PhysicsTeacher extends the designation and college properties and work() method from base class, we need not to declare these properties and method in sub class. Here we have collegeName, designation and work() method which are common to all the teachers so we have declared them in the base class, this way the child classes like MathTeacher, MusicTeacher and PhysicsTeacher do not need to write this code and can be used directly from base class. Write a sample to illustrate the above concept

VI

A)Write a method called printRange that accepts two integers as arguments and prints the sequence of numbers between the two arguments, enclosed in square brackets. Print an increasing sequence if the first argument is smaller than the second; otherwise, print a decreasing sequence. If the two numbers are the same, that number should be printed between square brackets. Here are some sample calls to printRange: printRange(2, 7); printRange(19, 11); printRange(5, 5);
The output produced from these calls should be the following sequences of numbers:
[2, 3, 4, 5, 6, 7] , [19, 18, 17, 16, 15, 14, 13, 12, 11] ,[5]

B)Write a Java class Author with following features:

- Instance variables :
  - **firstName** for the author's first name of type String and **lastName** for the author's last name of type String.
- Constructor:
  - **public Author (String firstName, String lastName)**: A constructor with parameters, it creates the Author object by setting the two fields to the passed values.
- Instance methods:
  - **public void setFirstName (String firstName)**: Used to set the first name of author.
  - **public void setLastName (String lastName)**: Used to set the last name of author.
  - **public double getFirstName()**: This method returns the first name of the author.
  - **public double getLastName()**: This method returns the last name of the author.

C) Create a class called Employee whose objects are records for an employee. This class will be a derived class of the class Person which you will have to copy into a file of your own and compile. An employee record has an employee's name (inherited from the class Person), an annual salary represented as a single value of type double, a year the employee started work as a single value of type int and a national insurancenumber, which is a value of type String. Your class should have a reasonable number of constructors and accessor methods, as well as an equals method. Write another class containing a main method to fully test your class definition.