Lab # 7

Writing Classes

Note:

- Please compile/run the following examples based on the instructions delivered in the lab.
- After learning the concepts successfully, complete the tasks provided at the end.

Example 1:

```
public class RollingDice
{
//-----------------------------------------------------------------
// Creates two Die objects and rolls them several times.
//-----------------------------------------------------------------
public static void main (String[] args)
{
Die die1, die2;
int sum;
die1 = new Die();
die2 = new Die();
die1.roll();
die2.roll();
System.out.println ("Die One: " + die1 + ", Die Two: " + die2);
die1.roll();
die2.setFaceValue(4);
System.out.println ("Die One: " + die1 + ", Die Two: " + die2);
sum = die1.getFaceValue() + die2.getFaceValue();
System.out.println ("Sum: " + sum);
sum = die1.roll() + die2.roll();
System.out.println ("Die One: " + die1 + ", Die Two: " + die2);
System.out.println ("New sum: " + sum);
}
}
```

```
public class Die
{
private final int MAX = 6; // maximum face value
private int faceValue; // current value showing on the die
//-----------------------------------------------------------------
// Constructor: Sets the initial face value.
//-----------------------------------------------------------------
public Die()
{
faceValue = 1;
}
//-----------------------------------------------------------------
// Rolls the die and returns the result.
//-----------------------------------------------------------------
public int roll()
{
faceValue = (int)(Math.random() * MAX) + 1;
return faceValue;
}
//-----------------------------------------------------------------
// Face value mutator.
//-----------------------------------------------------------------
public void setFaceValue (int value)
{
faceValue = value;
}
//-----------------------------------------------------------------
// Face value accessor.
//-----------------------------------------------------------------
public int getFaceValue()
{
return faceValue;
}
//-----------------------------------------------------------------
// Returns a string representation of this die.
//-----------------------------------------------------------------
```

```
public String toString()
{
String result = Integer.toString(faceValue);
return result;
}
}
```

Example 2:

```
public class Transactions
{
//-----------------------------------------------------------------
// Creates some bank accounts and requests various services.
//-----------------------------------------------------------------
public static void main (String[] args)
{
Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
Account acct2 = new Account ("Jane Smith", 69713, 40.00);
Account acct3 = new Account ("Edward Demsey", 93757, 759.32);
acct1.deposit (25.85);
double smithBalance = acct2.deposit (500.00);
System.out.println ("Smith balance after deposit: " +
smithBalance);
System.out.println ("Smith balance after withdrawal: " +
acct2.withdraw (430.75, 1.50));
acct1.addInterest();
acct2.addInterest();
acct3.addInterest();
System.out.println ();
System.out.println (acct1);
System.out.println (acct2);
System.out.println (acct3);
}
}
```

```
import java.text.NumberFormat;
public class Account
{
private final double RATE = 0.035; // interest rate of 3.5%
private long acctNumber;
private double balance;
private String name;
//-----------------------------------------------------------------
// Sets up the account by defining its owner, account number,
// and initial balance.
//-----------------------------------------------------------------
public Account (String owner, long account, double initial)
{
name = owner;
acctNumber = account;
balance = initial;
}
//-----------------------------------------------------------------
// Deposits the specified amount into the account. Returns the
// new balance.
//-----------------------------------------------------------------
public double deposit (double amount)
{
balance = balance + amount;
return balance;
}
//-----------------------------------------------------------------
// Withdraws the specified amount from the account and applies
// the fee. Returns the new balance.
//-----------------------------------------------------------------
public double withdraw (double amount, double fee)
{
balance = balance - amount - fee;
return balance;
}
//-----------------------------------------------------------------
```

```
// Adds interest to the account and returns the new balance.
public double addInterest ()
{
balance += (balance * RATE);
return balance;
}
// Returns the current balance of the account.
public double getBalance ()
{
return balance;
}
//-----------------------------------------------------------------
// Returns a one-line description of the account as a string.
//-----------------------------------------------------------------
public String toString ()
{
NumberFormat fmt = NumberFormat.getCurrencyInstance();
return acctNumber + "\t" + name + "\t" + fmt.format(balance);
}
}
```

Tasks to be performed in Lab

- Design and implement a class called *Sphere* that contains instance data that represents the sphere's diameter. Define the Sphere constructor to accept and initialize the diameter, and include *getter* and *setter* methods for the diameter. Include methods that calculate and return the volume and surface area of the sphere. Include a *toString* method that returns a one-line description of the sphere. Create a driver class called *MultiSphere*, whose main method instantiates and updates several Sphere objects.

- Design and implement a class called *Car* that contains instance data that represents the make, model, and year of the car. Define the Car constructor to initialize these values. Include *getter* and *setter* methods for all instance data, and a *toString* method that returns a one-line description of the car. Create a driver class called *CarTest*, whose main method instantiates and updates several Car objects.

- Using the *Die* class defined in example, design and implement a class called *PairOfDice*, composed of two Die objects. Include methods to set and get the

individual die values, a method to roll the dice, and a method that returns the current sum of the two die values. Create a driver class called *RollingDice2* to instantiate and use a *PairOfDice* object.

- Design and implement a class called *Flight* that represents an airline flight. It should contain instance data that represents the airline name, flight number, and the flight's origin and destination cities. Define the Flight constructor to accept and initialize all instance data. Include *getter* and *setter* methods for all instance data. Include a *toString* method that returns a one-line description of the flight. Create a driver class called *FlightTest*, whose main method instantiates and updates several Flight objects.

Tasks to be performed at home

Note: Bring the print out of these tasks in next lab.

- Design and implement a class called *Dog* that contains instance data that represents the dog's name and age. Define the Dog constructor to accept and initialize instance data. Include *getter* and *setter* methods for the name and age. Include a method to compute and return the age of the dog in "person years" (seven times the dogs age). Include a *toString* method that returns a one-line description of the dog. Create a driver class called *Kennel*, whose main method instantiates and updates several Dog objects.

- Design and implement a class called *Box* that contains instance data that represents the height, width, and depth of the box. Also include a boolean variable called full as instance data that represents whether the box is full or not. Define the Box constructor to accept and initialize the height, width, and depth of the box. Each newly created Box is empty (the constructor should initialize full to false). Include *getter* and *setter* methods for all instance data. Include a *toString* method that returns a oneline description of the box. Create a driver class called *BoxTest*, whose main method instantiates and updates several Box objects.

- Design and implement a class called *Book* that contains instance data for the title, author, publisher, and copyright date. Define the Book constructor to accept and initialize this data. Include *setter* and *getter* methods for all instance data. Include a *toString* method that returns a nicely formatted, multi-line description of the book. Create a driver class called *Bookshelf*, whose main method instantiates and updates several Book objects.