Exercise 6.1

1. Class is a static piece of code that consists of attributes which don’t change during the execution of a program. An object refers to an actual instance of a class.
2. Java assumes that it is okay to delete them from memory.
3. 1) behavior 2) identity 3) state
4. A client’s interactions with a server are limited to sending it messages, so consequently a client needs to know nothing about the internal workings of a server.
5. Contain only constants, method signatures, default methods, static methods, and nested

Exercises 6.2

1. Mutators are messages that change an object’s state. Accessors are messages to access the object’s state. M: s1.setName (“Bill”); A: str = s1.getName();
2. The private modifier is an important aspect of information hiding and the public modifier allows clients to refer to the methods.
3. A method that initializes the instance variables of a newly instantiated object.
4. To send the user the output of the program.
5. That is how Java works. (s1 already defined) s2 = s1; //now refer to same obj
6. Primitive type is a like a box that contains a value of that type, a reference type is a box that contains a pointer to an object.
7. A value that clears the variable’s contents.
8. When a value is null and something tries to access it. String str = null; out.print(str.length());
9. Initializes the variables to zero or empty strings.
10. JVM creates a default constructor.
11. Setting the object to null...??
12. types. Method bodies exist only for default methods and static methods

Exercise 6.4

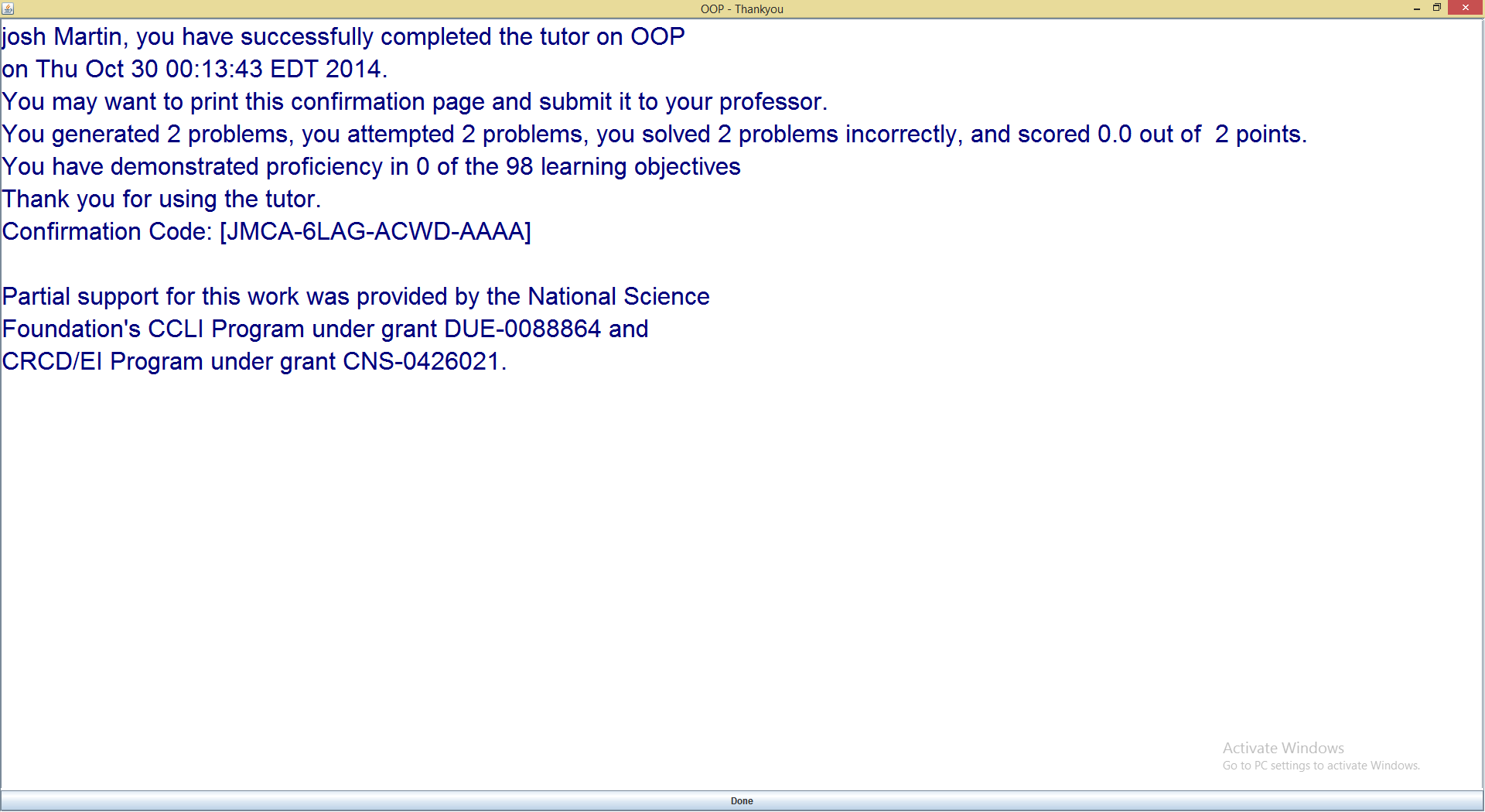
1. Formal are parameters listed in a method’s definition. An actual is when values passed to method.
2. When method is called, the value of the parameter is automatically transferred to the corresponding formal parameter immediately before the method us activated.
3. Mi no Gusta :/
4. To have temporary working storage for data in a method.

Exercise 6.5

1. Once a method stops executing, the parameters and local variables are no longer accessible. Instance variables last the lifetime of an object.
2. The instance variable with the same name as a local variable. Shadowing is considered a dangerous programming practice because it greatly increases the like hood of making a coding error.
3. A. a,b, x,y,c,d

B. a & b scope is the whole program. x,y,c,d is only inside of the amutator method.

C. A & b forever, x& y lifetime of the method, and c & d





**import** java.util.Scanner;

//project 6-3

**public** **class** Fraction {

**private** **int** num,denom;

**public** Fraction()

{

num= 0;

denom = 1;

}

**public** Fraction (**int** n, **int** d)

{

num= n;

denom = d;

}

**public** **int** getNum()

{

**return** num;

}

**public** **int** getDenom()

{

**return** denom;

}

// the real work

**public** Fraction add(Fraction other)

{

**int** newNum,newDenom;

**if** (denom == other.denom)

{

newNum= num + other.num;

newDenom= denom;

}

**else**

{

newNum = num\*other.denom+ other.num \* **this**.denom;

newDenom = **this**.denom \* other.denom;

}

**return** **new** Fraction();

}

**public** Fraction subtract(Fraction other)

{

**int** newNum,newDenom;

**if** (denom == other.denom)

{

newNum= num - other.num;

newDenom= denom;

}

**else**

{

newNum = num\*other.denom - other.num \* **this**.denom;

newDenom = **this**.denom \* other.denom;

}

**return** **new** Fraction();

}

**public** Fraction mulitply(Fraction other)

{

**int** newNum,newDenom;

**if** (denom == other.denom)

{

newNum= num \* other.num;

newDenom= denom;

}

**else**

{

newNum = num\*other.num;

newDenom = **this**.denom \* other.denom;

}

**return** **new** Fraction();

}

**public** Fraction divide(Fraction other)

{

**int** newNum,newDenom;

newNum = num\*other.denom;

newDenom = **this**.denom \* other.num;

**return** **new** Fraction();

}

**public** String toString()

{

**return** num +"/" + denom;

}

}// end of project class

//project 6-3

**public** **class** fractiontester {

**public** **static** **void** main(String args[])

{

Fraction f1,f2,f3,f4,f5,f6,f7,f8;

f1 = **new** Fraction();

f2 = **new** Fraction(2,3);

f3 = **new** Fraction(3,4);

f4 = **new** Fraction(4,3);

System.out.println(f2.getNum());

System.out.println(f3.getDenom());

f5 = f1.add(f2);

f6 = f2.subtract(f3);

f7 =f3.mulitply(f4);

f8 = f4.divide(f1);

System.out.println(f5);

System.out.println(f6);

System.out.println(f7);

System.out.println(f8);

}

}

Critical thinking Pg.228

1. You could check if (top/bottom)% 2 == 0

//project6-4

**import** java.util.Scanner;

**import** java.util.Random;

**public** **class** LuckySevens {

**private** **int** dollars, maxd, counterols, highmoney;

**public** LuckySevens(**int** a) {

dollars = a;

roll();

}

**public** **void** roll(){

**int** die1, die2,

count = 0,

maxDollars = dollars,

countAtMax = 0;

Random generator = **new** Random();

**while** (dollars > 0){

count++;

// Roll the dice.

die1 = generator.nextInt (6) + 1; // 1-6

die2 = generator.nextInt (6) + 1; // 1-6

// Calculate the winnings or losses

**if** (die1 + die2 == 7)

dollars += 4;

**else**

dollars -= 1;

// If this is a new maximum, remember it

**if** (dollars > maxDollars){

maxDollars = dollars;

countAtMax = count;

}

}

maxd = maxDollars; counterols = count; highmoney = countAtMax;

}

**public** String toString()

{

**return** "You are broke after " + counterols + " rolls.\n" + "You should have quit after " + highmoney + " rolls when you had $" + maxd + ".";

}

Project 6-6 p. 227-228

Exercise 7.1



|  |  |  |
| --- | --- | --- |
| P | Q | !((P|| Q) && (P && Q)) |
| T | T | false |
| T | F | true |
| F | T | true |
| F | F | true |

1. A. True

B. false

C. true

D. False

1. Djakdhakjhfdajh djhjf
2. (), . , + , - , ! , \* , / , % , + , - , < > <= >= , || , = \*= /= %=
3. Table I have no idea how to do. Why do I not do anything in here?

Exercise 7.3

|  |  |  |
| --- | --- | --- |
| The time is before noon | The day is Monday | Action Taken |
| True | True | Take the computer science quiz |
| True | False | Go to gym class |
| False | False | Throw a Frisbee in the quad |

1. Nested :

If ( x >1 )

//Code goes here

If (x<2)

//Code goes here

else

//Code goes here

Else

//Code goes here

Multiway:

If ( x >1 )

//Code goes here

Else If (x<2)

//Code goes here

Else

//Code goes here

Exercise 7.4

1. Change > to =>
2. If (income => 1000)

Rate = .10;

Else if ( income >1000 && income =< 2000)

Rate = .18;

Else if ( income => 5000)

Rate = .40;

Else

Rate = 0.0;

Exercise 7.5

1. 1 2 3 1 2 3 1 2 3

1 2 3

1 2 3

1 2 3

1. A.

b.

Exercise 7.8 p. 269-70

1. A. 3 b. 8 c. rain in Spain falls mainly on the plain d. rain\_ e. idk
2. Str = str.replace( ‘ ‘ , ‘\n’) b. index = str.indexof(‘the’) c.mid = str.indexof(‘the’, str.length()/2) d. dkfjaklhj

Ex 8.4 p. 292

1. A. system.out.printf(“%s”,”One Space”) b. system.out.printf(“%2s”, ”Two Space”) c. system.out.printf(“s%3”, ”three Space”) d. e.
2. A. Price 10000.50

b. 45 632

c. 34.54

Exercise 8.5 p. 294

1. The computer is going to hit an error because $12.55 is a string
2. The statements within the try clause are executed until one of them throws an exception. If that happens, an exception object is created and sent immediately to the catch clause
3. ffgaaa

