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 types. Method bodies exist only for default methods and static methods

Exercise 6.4

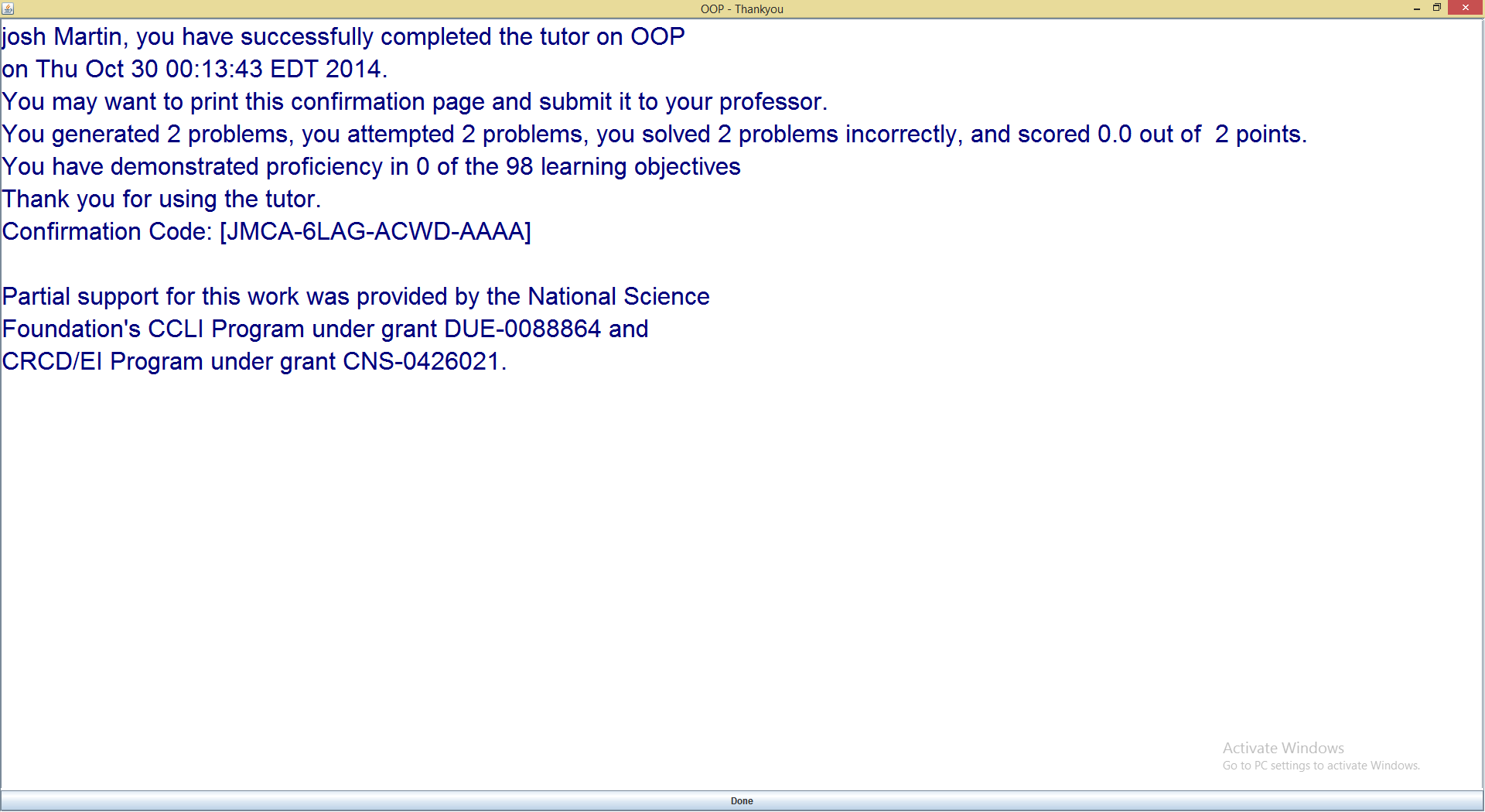
1. Formal are parameters listed in a method’s definition. An actual is when values passed to method.
2. When method is called, the value of the parameter is automatically transferred to the corresponding formal parameter immediately before the method us activated.
3. Mi no Gusta :/
4. To have temporary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

}

