What are methods?

- We have seen a few examples of procedures (in Java, we call them methods):
  - `System.out.println`, which we have used to display output on the screen
  - `Keyb.nextInt`, which we have used to get integer inputs from the keyboard
  - `newRandomNumber.nextInt()`, which we have used to get a random numbers
- Functions allow us to use software routines that have already been written (frequently by other people) in our programs.
  E.g., `magic = newRandomNumber.nextInt();`
Why use methods?

- Methods offer several advantages when we write programs:
  - They allow us to concentrate on a higher level abstractions, without getting bogged down in details that we are not yet ready to handle.
  - They make it easier to divide the work of writing a program among several people.
  - They are re-usable; i.e., we write it once and can use it several times in a program and we can even copy it from one program to another.

Simple Methods to print messages

- Let’s start with a simple function: Let’s a function that will print instructions for a user playing the “Magic Number” game:

```java
// printInstruction() - Print instructions for the user
public static void printInstructions() {
    System.out.println("The object of the game is "+ "to find out");
    System.out.println("which number the computer "+ "has picked. The");
    System.out.println("computer will tell you if "+ "you guessed too");
    System.out.println("high a number or too low. "+ "Try to get it with");
    System.out.println("as few guesses as possible.\n");
}
```
Simple Functions For Printing Messages

• The general form of the syntax is:

  ```java
  public static void FunctionName(void)
  {
    Statement(s)
  }
  ```

Putting the Pieces Together

```java
import java.util.*;

public class MagicNumber2 {
    // The magic number game has the user trying
to guess
    // which number between 1 and 100 the computer
has picked
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        Random newRandomNumber = new Random();
        int magic, guess;
        int tries = 1;

        printInstructions();
        // Use the random number function to pick a
        // number
        magic = newRandomNumber.nextInt(100) + 1;
    }
}
```
do {
    // Let the user make a guess
    System.out.println("Guess ?");
    guess = keyb.nextInt();

    // If the user won, tell him/her
    if (guess == magic) {
        System.out.println("** Right!! ** ");
        System.out.println(magic +
            " is the magic number\n");
    }

    // Otherwise tell him whether it's
too high or too low
    else if (guess > magic)
        System.out.println(".. Wrong .. Too high\n");
    else
        System.out.println(".. Wrong .. Too low\n");

    // Let the user make another guess
    tries++;
} while (guess != magic);

// Tell the user how many guesses it took
System.out.println("You took "+ tries + " guesses\n");
// printInstruction() - Print instructions for
// the user
public static void printInstructions() {
    System.out.println("The object of the game is to find out");
    System.out.println("which number the computer has picked.");
    System.out.println("The computer will tell you if you ");
    System.out.println("guessed too high a number or too low. " + "Try to get it with");
    System.out.println("as few guesses as possible.\n\n");
}

Another Function to Print Messages

import java.util.Scanner;

public class CalcGrade2 {
    // Calculates the average test grade and converts it to a letter grade assuming that
    // A is a 90 average, B is an 80 average and so on.
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        final int sentinelGrade = -1;
        int thisGrade, numTests = 0, total = 0;
        float testAverage;
        char courseGrade;

        // First, print the instructions
        printInstructions();
    }
}
// Get the first grade
System.out.println("What grade did you get on your "+" first test ?");
System.out.println("Enter -1 to end");
thisGrade = keyb.nextInt();

// Add up the test grades
while (thisGrade != sentinelGrade) {
    // Make sure that the grades are valid
    // percentages
    if (thisGrade > 100)
        System.out.println("This is not a valid test grade.");
    else if (thisGrade >= 0) {
        total = total + thisGrade;
        numTests++;
    } else
        System.out.println("This is not a valid test grade.");

    System.out.println("What grade did you get on this test?";
    thisGrade = keyb.nextInt();
}
// Find the average
testAverage = total/numTests;
// Find the letter grade corresponding to the average
if (testAverage >= 90)
    courseGrade = 'A';
else if (testAverage >= 80)
    courseGrade = 'B';
else if (testAverage >= 70)
    courseGrade = 'C';
else if (testAverage >= 60)
    courseGrade = 'D';
else
    courseGrade = 'F';

// Print the results.
System.out.println("Your test average is 
    + testAverage);
System.out.println("Your grade will be 
    + courseGrade);
}

// printInstructions() - Print instructions for the user
public static void printInstructions() {
    // Print an introductory message
    System.out.println("This program calculates" + " your test average");
    System.out.println("assuming that all tests have the 
    + "same weight in your grade.");
    System.out.println("It also assumes that averages 
    + "in the 90s are As, in the 80s");
    System.out.println("are Bs and so on.
    ");
    System.out.println("To indicate that you are finished," + " enter a grade of -1
    ");
}
What are parameters?

• A **parameter** is a value or a variable that is used to provide information to a function that is being called.

• If we are writing a function to calculate the square of a number, we can pass the value to be squared as a parameter:

  ```java
  printSquare(5);
  printSquare(x)
  ```

• These are called actual parameters because these are the actual values (or variables) used by the function being called.

---

Formal Parameters

• Functions that use parameters must have them listed in the function header. These parameters are called **formal parameters**.

  ```java
  public static void printSquare(double x) {
      double square;
      square = x*x;
      System.out.println("The square of "+ x + " is "+ square);
  }
  ```
Parameter Passing

\[
\text{printSquare}(5); \quad \text{printSquare}(x)
\]

\[
\text{public static void printSquare(double x)}
\]
\[
\quad \{ \\
\quad \quad \text{double square;} \\
\quad \quad \text{square} = \text{x} \times \text{x}; \\
\quad \quad \text{System.out.println("The square of " + x + " is " + square);} \\
\quad \}\n\]

In both cases, calling the function requires copying the actual parameter's value where the function can use it. Initially, \(x\) has whatever value the actual parameter has.

Parameter Passing (continued)

\[
\text{printSquare}(5)
\]

\[
\text{public static void printSquare(double x)}
\]
\[
\quad \{ \\
\quad \quad \text{double square;} \\
\quad \quad \text{square} = \text{x} \times \text{x}; \\
\quad \quad \text{System.out.println("The square of " + x + " is " + square);} \\
\quad \}\n\]

\[
\text{x initially is set to 5. square is then set to the value of } x^2 \text{ or } 5^2 \text{ or 25.}
\]
Parameter Passing (continued)

```java
public static void printSquare(double x)
{
    double square;
    square = x*x;
    System.out.println("The square of "+ x + " is " + square);
}
```

*x initially is set to whatever value x had in the main program. If x had the value 12, square is then set to the value of \( x^2 \) or \( 12^2 \) or 144.*

Why parameters?

- Parameters are useful because:
  - They allow us to use the same function in different places in the program and to work with different data.
  - They allow the main program to communicate with the function and pass it whatever data it is going to use.
  - The same value can have completely different names in the main program and in the function.
import java.util.Scanner;

public class Squares {

    // main() - A driver for the print_square function
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        double value;

        // Get a value and print its square
        System.out.println("Enter a value ?");
        value = keyb.nextDouble();
        printSquare(value);
    }

    // printSquare() - Prints the square of whatever value that it is given.
    public static void print_square(double x) {
        double square;
        square = x*x;
        System.out.println("The square of " + x + " is " + square);
    }

}
Passing Parameters - When The User Inputs 12

Value: 12 → 12
Square: 144

Passing Parameters - When The User Inputs 6

Value: 6 → 6
Square: 36
import java.util.Scanner;

public class Squares2 {
    // main() - A driver for the print_square function
    public static void main(String[] args) {
        double value1 = 45, value2 = 25;

        printSquare(value1);
        printSquare(value2);
    }
}

Passing Parameters - Using square Twice In One Program

Value1: 45
Value2: 25

Value1: 45
Value2: 25

45
2025

6
36
A program to calculate Grade Point Average

**Example** - Ivy College uses a grading system, where the passing grades are A, B, C, and D and where F (or any other grade) is a failing grade. Assuming that all courses have equal weight and that the letter grades have the following numerical value:

<table>
<thead>
<tr>
<th>Letter grade</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
</tr>
</tbody>
</table>

write a program that will calculate a student's grade point average.

Let’s Add– Dean’s List

- Let’s include within the program a method that will print a congratulatory message if the student makes the Dean’s List.
- We will write a function `deansList` that will print the congratulatory message and another method `printInstructions`. 
A program to calculate Grade Point Average

**Input** - The student's grades
**Output** - Grade point average and a congratulatory message (if appropriate)

**Other information**
"A" is equivalent to 4 and so on
GPA = Sum of the numerical equivalents/ Number of grades

Our first step is to write out our initial algorithm:

1. Print introductory message
2. Add up the numerical equivalents of all the grades
3. Calculate the grade point average and print it out
4. Print a congratulatory message (if appropriate)

Refining the GPA Algorithm

1. Print introductory message
2. Add up the numerical equivalents of all the grades
3. Calculate the grade point average and print it out
4. Print a congratulatory message (if appropriate)

printInstructions();
Refining the GPA Algorithm

printInstructions();

2. Add up the numerical equivalents of all the grades
3. Calculate the grade point average and print it out
4. Print a congratulatory message (if appropriate)

2.1 Get the first grade
2.2 While the grade is not X:
2.3 Add the numerical equivalent to the total
2.4 Get the next grade

3.1 Calculate Gpa = Point total / Number of courses
3.2 Print the Gpa
Refining the GPA Algorithm

printInstructions();
2.1 Get the first grade
2.2 While the grade is not X:
2.3 Add the numerical equivalent to the total
2.4 Get the next grade
3.1 Calculate Gpa = Point total / Number of courses
3.2 Print the Gpa
4. Print a congratulatory message (if appropriate)
deansList(gpa)

IF Grade = 'A' THEN Add 4 to Total
ELSE IF Grade = 'B' THEN Add 3 to Total
ELSE IF Grade = 'C' THEN Add 2 to Total
ELSE IF Grade = 'D' THEN Add 1 to Total
The `deanLists()` method

IF gpa >= 3.2
   Print congratulatory message

---

The Entire `DeansList` Program

```java
import java.util.Scanner;

public class DeansList {
   // Calculates a grade point average assuming
   // that all courses have the same point value
   // and that A, B, C and D are passing grades and
   // that all other grades are failing.
   public static void main(String[] args) {
      Scanner keyb = new Scanner(System.in);
      int numCourses = 0;
      char grade;
      String inputString = new String();
      double gpa, total = 0;
      
      printInstructions();
   }
}
```
// Get the first course grade
System.out.println("What grade did you get in" + " your first class?");
inputString = keyb.next();
grade = inputString.charAt(0);

// Add up the numerical equivalents of the grades
while (grade != 'X') {
    // Convert an A to a 4, B to a 3, etc.
    // and add it to the total
    if (grade == 'A')
        total = total + 4;
    else if (grade == 'B')
        total = total + 3;
    else if (grade == 'C')
        total = total + 2;
    else if (grade == 'D')
        total = total + 1;
    else if (grade != 'F')
        System.out.println("A grade of " + grade + " is assumed to be an F\n");
    numCourses++;
}
// Get the next course grade
System.out.println
    ("What grade did you get in the" + " next class?");
inputString = keyb.next();
grade = inputString.charAt(0);
}

// Divide the point total by the number of
// classes to get the grade point average
// and print it.
gpa = total / numCourses;
System.out.printf
    ("Your grade point average is %4.2f\n", gpa);
dehnsList(gpa);

// printInstructions() - Prints instructions
//                       for the user
public static void printInstructions() {
    // Print an introductory message
    System.out.println
        ("This program calculates your grade point" + " average\n" + "assuming that all courses have the same" + "point \n");
    System.out.println
        ("value. It also assumes that grades of " + "A, B, C and D\n");
    System.out.println
        ("are passing and that all other grades " + "are failing.\n");
    System.out.println
        ("To indicate that you are finished, " + "enter a grade of \'X\n\n");
}
Calculating the Average of 3 Values Using a Function

- Let’s re-examine how to find the average of 3 values. We have to:
  1. Get the values as input
  2. Calculate and display the average

- We can refine step #1:
  1.1 Get value 1
  1.2 Get value 2
  1.3 Get value 3
  2. Calculate and display the average
Refining the Average of 3 Values Algorithm

- We can further refine steps #1.1, 1.2, 1.3:
  1.1.1 Prompt the user for value 1
  1.1.2 Get value 1
  1.1.1 Prompt the user for value 2
  1.1.2 Get value 2
  1.1.1 Prompt the user for value 3
  1.1.2 Get value 3

2. Calculate and display the average

```java
System.out.println("Enter a value ?");
value1 = keyb.nextInt();

System.out.println("Enter a value ?");
value2 = keyb.nextInt();
System.out.println("Enter a value ?");
value3 = keyb.nextInt();
```

2. Calculate and display the average
Refining the Average of 3 Values Algorithm

- We can further refine steps #1.1, 1.2, 1.3:

```java
System.out.println("Enter a value ");
value1 = keyb.nextInt();
System.out.println("Enter a value ");
value2 = keyb.nextInt();
System.out.println("Enter a value ");
value3 = keyb.nextInt();
```

2. Calculate and display the average

```java
findAverage(value1, value2, value3);
```

---

The `average3` Program

```java
import java.util.Scanner;

public class Average3Func {

    // Find the average of three numbers using a function
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        int value1, value2, value3;
        double average;

        //Get the inputs
        System.out.println("Enter a value ");
        value1 = keyb.nextInt();
        System.out.println("Enter a value ");
        value2 = keyb.nextInt();
        System.out.println("Enter a value ");
        value3 = keyb.nextInt();
```
System.out.println("Enter a value ");
value3 = keyb.nextInt();

// Call the function that calculates and
// prints the average
printAverage(value1, value2, value3);
}

// find_average() – Find the average of three
// numbers
public static void printAverage(int x,
int y, int z) {
    double sum, average;
    sum = x + y + z;
    average = sum / 3;
    System.out.printf("The average is %3.1f\n", average);
}

Example – x to the nth power

• Let’s write a function to calculate x to the
nth power and a driver for it (a main
program whose sole purpose is to test the
function.

• Our basic algorithm for the function:
  – Initialize (set) the product to 1
  – As long as n is greater than 0:
    • Multiply the product by x
    • Subtract one from n
import java.util.Scanner;

public class Power {
    // A program to calculate 4-cubed using a
    // function called power
    public static void main(String[] args) {
        double x, y;
        int n;

        x = 4.0;
        n = 3;
        y = 1.0;
        power(y, x, n);
        System.out.println("The answer is " + y);
    }

    // power() - Calculates y = x to the nth power
    public static void power(double y,
                              double x, int n) {
        y = 1.0;
        while (n > 0) {
            y = y * x;
            n = n - 1;
        }
        System.out.println("Our result is " + y);
    }
}
The Output From \texttt{power}

Our result is 64
The answer is 1

\begin{quote}
Shouldn’t these be the same numbers?
\end{quote}

The problem is that communication using parameters has been one-way – the function being called listens to the main program, but the main program does not listen to the function.

Value Parameters

- The parameters that we have used all pass information from the main program to the function being called by copying the values of the parameters. We call this \textit{passing by value}, because the value itself is passed.
- Because we are using a copy of the value copied in another location, the original is unaffected.
Methods and Functions

• Some methods perform specific tasks and do not produce any one data item that seem to be their whole reason for existence.

• Other methods are all about producing some value or data item; in many programming languages they are called *functions*.

**void** Functions

• Normally a function is expected to produce some **result** which is returned to the **main** program:
  
  ```java
  average = calcAverage(x, y, z);
  ```

• The data type of the function’s result is also called the function’s type.
  – Functions that produce an integer are called **integer functions**.
  – Functions that produce float value are called **float functions**.
  – Functions that do not produce a result are called **void functions**.

• When we write
  
  ```java
  public void printSquare(int x);
  ```
  
  it means that the function is not expected to return a result.
Writing Functions That Return Results

• We can write a function that returns a result by replacing that void with a data type:
  `public double average3(int a, int b, int c);`
• The rest of the function is a little different from before:

```java
public double average3(int a, int b, int c) {
    float sum, mean;
    sum = a + b + c;
    mean = sum / 3;
    return mean;
}
```

The result that we are returning is mean

Writing Functions That Return Results

• The syntax is:
  `return expression;`
• Return statements have contain expressions, variables, constants or literals:
  `return true;`
  `return 35.4;`
  `return sum;`
  `return sum/3;`
Rewriting the `average3` Function

```java
public double average3(int a, int b, int c)
{
    float sum, mean;
    sum = a + b + c;
    return sum / 3;
}
```

Maximum and Minimum

- Let’s write a pair of functions that find the minimum and maximum of two values `a` and `b`.
- Initial algorithm for maximum:
  Return the larger of `a` and `b`:
  - If we refine this:
    1.1 IF `a` > `b` return `a`
    1.1 else return `b` //`a` ≤ `b`

- For minimum, we replace > with <
public double maximum(float x, float y) {
    if (x > y)
        return(x);
    else
        return(y);
}

public double minimum(float x, float y) {
    if (x < y)
        return(x);
    else
        return(y);
}
import java.util.Scanner;

public class Payroll3 {

    // A simple payroll program that uses a method
    // to calculate the gross pay
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        double hours, rate, pay;

        // Ask the user for payrate
        System.out.println("What is rate of pay for the employee?" );
        rate = keyb.nextDouble();

        // Enter the hours worked
        System.out.println("Enter the hours worked?");
        hours = keyb.nextInt();

        // Get the gross pay
        pay = gross(hours, rate);
        System.out.printf("Gross pay is $%4.2f\n",pay);
    }

    // gross() - Calculate the gross pay.
    public static double gross(double hours, double rate) {
        double pay;

        // If hours exceed 40, pay time and a half
        if (hours > 40)
            pay = 40*rate + 1.5*rate*(hours-40);
        else
            pay = rate * hours;
        return pay;
    }
}
return

• return serves two purposes:
  – It tells the computer the value to return as the result.
  – It tell the computer to leave the function immediately and return the main program.

// gross() – Calculate the gross pay.
public static double gross(double hours, double rate) {
   // If hours exceed 40, pay time and a half
   if (hours > 40)
      return(40*rate + 1.5*rate*(hours-40));
   return(rate*hours);
}

Rewriting pow

• We can make the pow function tell the main program about the change in \( y \) by having it return the value as the result:

public static double power(double x, int n) {
   ...
   }

The rewritten **pow** program

```java
import java.util.Scanner;

public class PowerTest {
    // A program to calculate 4-cubed using a
    // function called power
    public static void main(String[] args) {
        double x, y;
        int n;

        x = 4.0;
        n = 3;
        y = power(x, n);
        System.out.println("The answer is " + y);
    }

    // power() - Calculates y = x to the nth
    // power
    public static double power(double x, int n) {
        double prod;

        prod = 1.0;
        while (n > 0) {
            prod = prod * x;
            n = n - 1;
        }
        System.out.println("Our result is " + prod);
        return prod;
    }
}
```
The New Output From \texttt{power}

Our result is 64
The answer is 64

Exactly what we would expect Why?

Communication using the result is two-way – the function being called listens to the main program, but the main program listens to the function because data changes are explicitly passed back to the main method.

An Example – \texttt{square2}

- Let’s rewrite the square program so that the function calculates the square and passes its value back to the main program, which will print the result:

```java
import java.util.Scanner;

public class Square2 {
    // This illustrates how to use methods to find the square of a value

    // main() – A driver for the findSquare method
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        double value, square;

        System.out.println("Enter a value ?");
        value = keyb.nextDouble();
    }
```
square = findSquare(value);
System.out.println("The square of " + value
 + " is " + square);
}

// findSquare() - Calculates the square of
// whatever value it is given.
public static double findSquare(double x) {
    double square = x*x;
    return square;
}
}

Comparing print_square and find_square

• What are the differences between print_square and find_square?

• print_square:
  – uses value parameters
  – prints the square; it doesn’t have to pass that
    value to the main program

• find_square:
  – Returns the result
  – does not print the square; it must pass the value
    back to the main program
Example: Average3

- Let’s write a program which will find the average of three numbers:
- Our algorithm is:
  1. Read the values
  2. Calculate the average
  3. Print the average

Refining average3’s algorithm

1. Read the values
2. Calculate the average
3. Print the average

  1.1 Get value1
  1.2 Get value2
  1.3 Get value3
Refining average3’s algorithm (continued)

1.1 Get value1
1.2 Get value2
1.3 Get value3

- Calculate the average
- Print the average

```java
value1 = getValue();
value2 = getValue();
value3 = getValue();
```

Refining average3’s algorithm (continued)

```java
value1 = getValue();
value2 = getValue();
value3 = getValue();
```

2. Calculate the average
3. Print the average

```java
average = findAverage(value1, value2, value3);
```
Refining `average3`’s algorithm (continued)

```java
getValue(value1);
getValue(value2);
getValue(value3);
average = findAverage(value1, value2, value3);
Print the average
System.out.println("The average is " + average);
```

```
Average3c.java
import java.util.Scanner;
public class Average3c {
   // Find the average of three numbers using a
   // function
   public static void main(String[] args) {
      int value1, value2, value3;
      double average;

      //Get the inputs
      value1 = getValue();
      value2 = getValue();
      value3 = getValue();

      // Call the function that calculates the
      // average
      average = findAverage(value1, value2, value3);
      System.out.println("The average is " + average);
   }
}
```
// getValue() - Prompt the user and read a value
public static int getValue() {
    Scanner keyb = new Scanner(System.in);
    System.out.println("Enter a value ?");
    int x = keyb.nextInt();
    return x;
}

// find_average() - Find the average of three numbers
public static double findAverage(int x, int y, int z) {
    double sum = x + y + z;
    double average = sum / 3;
    return average;
}

Revising the **Nim** program

- Let’s revise the Nim program to use functions.
- We’ll create the following functions to subdivide the work:
  - `print_instructions`
  - `get_move`
  - `plan_move`
  - `update_sticks`
import java.util.Scanner;

public class Nim2 {

    // Play the game Nim against the computer
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        int sticksLeft = 7, pickUp = 0, reply = 0;
        boolean winner = false, move;
        char answer = ' ';
        String answerString = new String();

        printInstructions();

        // Find out if the user wants to go first or second
        while (answer != 'f' && answer != 'F'
            && answer != 's' && answer != 'S') {
            System.out.println
                ("Do you wish to go (f)irst or"
                + " (s)econd ?");
            answerString = keyb.next();
            answer = answerString.charAt(0);
        }

        // If the user goes second, have the computer take two sticks.
        if (answer == 's' || answer == 'S') {
            reply = 2;
            sticksLeft = sticksLeft - reply;
            System.out.println("The computer took "
                + reply + " sticks leaving "
                + sticksLeft + " on the table.");
        }
    }
}

// Find out if the user wants to go first or second
// While (answer != 'f' && answer != 'F'
//     && answer != 's' && answer != 'S') {
//    System.out.printn
//        ("Do you wish to go (f)irst or"
//        + " (s)econd ?");
//    answerString = keyb.next();
//    answer = answerString.charAt(0);
//}

// If the user goes second, have the computer take two sticks.
// take two sticks.
if (answer == 's' || answer == 'S') {
    reply = 2;
    sticksLeft = sticksLeft - reply;
    System.out.println("The computer took "
        + reply + " sticks leaving "
        + sticksLeft + " on the table.");
}
// If the user goes first, tell him how many
// sticks are on the table
else
    System.out.println("There are "+sticksLeft
    + " on the table.");

// As long as there is no winner, keep playing
while (!winner) {
pickUp = getMove(keyb, sticksLeft);

    // Take the sticks off the table
    sticksLeft = sticksLeft - pickUp;

    // See if the user won
    if (sticksLeft == 1) {
        System.out.println
            ("Congratulations!  You won!");
        winner = true;
    }

    // See if the user lost
    else if (sticksLeft == 0) {
        System.out.println
            ("Sorry, the computer has "
            + "won - you have lost..."降到
        winner = true;
    }
    else
        reply = planMove(sticksLeft);
if (!winner)
    sticksLeft
        = updateSticks(sticksLeft, reply);
}
// printInstructions() - Print instructions for the player
public static void printInstructions() {
    // Print the instructions
    System.out.println("There are seven (7) sticks on the table.");
    System.out.println("Each player can pick up one, two, or "+ "three sticks");
    System.out.println("in a given turn. A player cannot pick "+ "up more than");
    System.out.println("three stick nor can a player pass.\n");
}

// getMove() - Get the player's next move, testing to ensure that it is legal and that there are enough sticks on the table.
public static int getMove(Scanner keyb, int sticksLeft) {
    int pickUp = 0;
    boolean move = false;

    // How many sticks is the user taking
    while (!move) {
        System.out.println("How many sticks do you " + "wish to pick up?");
        pickUp = keyb.nextInt();
        System.out.println("Okay, I will add " + pickUp + " to the total.");
    }

    return pickUp;
}
// Make sure its 1, 2 or 3
if (pickUp < 1 || pickUp > 3)
    System.out.println(pickUp + " is not a legal number of sticks");
//Make sure that there are enough sticks on
// the table
else if (pickUp > sticksLeft)
    System.out.println("There are not " + pickUp + " sticks on the table");
else
    move = true;
}
return pickUp;

// planMove() - Plan the computer's next move
public static int planMove(int sticksLeft) {
    int reply = 0;

    // Plan the computer's next move
    if (sticksLeft == 6 || sticksLeft == 5 || sticksLeft == 2)
        reply = 1;
    else if (sticksLeft == 4)
        reply = 3;
    else if (sticksLeft == 3)
        reply = 2;
    return reply;
}
public static int updateSticks(int sticksLeft, int reply) {
    // If neither player won, get ready for the next move
    sticksLeft = sticksLeft - reply;
    System.out.println("The computer picked up "+ reply + " sticks.");
    System.out.println("There are now "+ sticksLeft + " sticks left on the table.
    " + sticksLeft + " sticks left on the table.");
    return sticksLeft;
}

Preconditions and Postconditions

• Preconditions – are conditions that we expect and require to be true before entering the procedure
• Postconditions – are conditions that we expect and require to be true after exiting the procedure
• Examples in square3:
  – getinput has a postcondition that a value was read in and that the value is set.
  – find average has a precondition that all value1, value2 and value al have values.