COMP 202
Java in one week

<u>CONTENTS</u>:
Basics of Programming
Variables and Assignment
Data Types: int, float, (string)
Example: Implementing a calculator

The Java Programming Language

- A *programming language* specifies the words and symbols that we can use to write a program

- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*

- Java was created by Sun Microsystems, Inc.
- It was introduced in 1995 and has become quite popular
- It is an object-oriented language

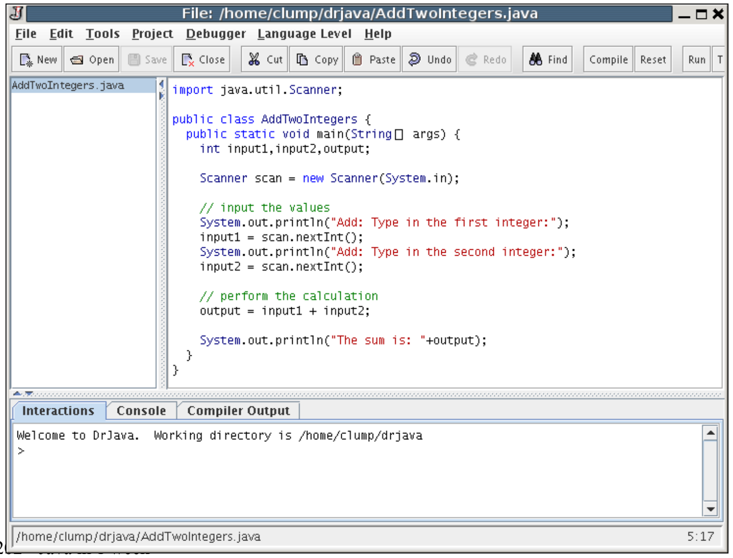
Java Program Structure

- In the Java programming language:
  – A program is made up of one or more *classes*
  – A class contains one or more *methods*
  – A method contains program *statements*
  – Statements are the actual commands you issue
- These terms will be explored in detail throughout the course

- A Java program always contains a method called `main`

This is where the program starts

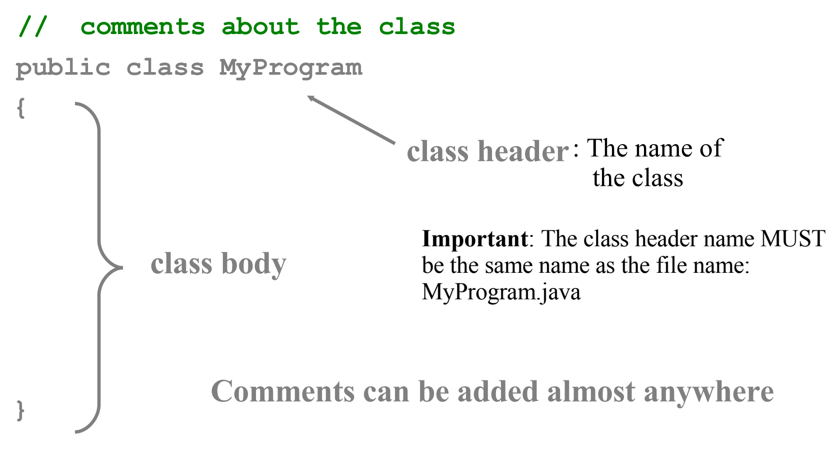
Calculator I: add two Integers


File: /home/clump/drjava/AddTwoIntegers.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset Run T
AddTwoIntegers.java
Interactions Console Compiler Output
/home/clump/drjava/AddTwoIntegers.java
5:17

```
import java.util.Scanner;

public class AddTwoIntegers {
  public static void main(String[] args) {
    int input1,input2,output;

    Scanner scan = new Scanner(System.in);

    // input the values
    System.out.println("Add: Type in the first integer:");
    input1 = scan.nextInt();
    System.out.println("Add: Type in the second integer:");
    input2 = scan.nextInt();

    // perform the calculation
    output = input1 + input2;

    System.out.println("The sum is: "+output);
  }
}
```

```
Welcome to DrJava.  Working directory is /home/clump/drjava
>
```

Java Program Structure

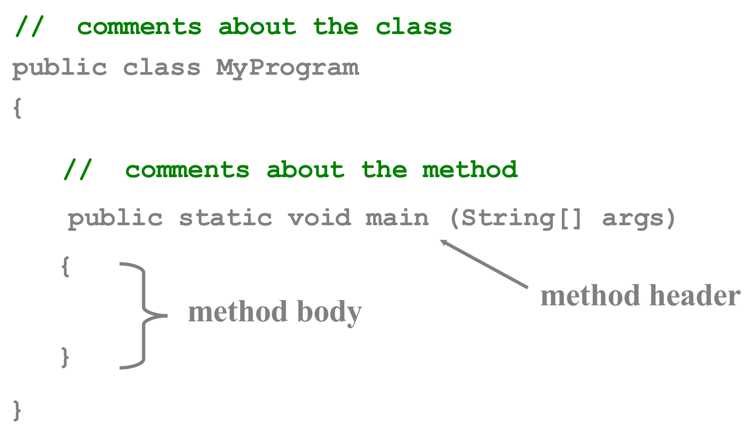


```
//  comments about the class
public class MyProgram
{

}
```

class header: The name of the class
class body
Comments can be added almost anywhere

Important: The class header name MUST be the same name as the file name: MyProgram.java

Java Program Structure



```
//  comments about the class
public class MyProgram
{

    //  comments about the method
    public static void main (String[] args)
    {

    }
}
```

Identifiers

- *Identifiers* are the words a programmer uses in a program. They are used to give names to things.
- An identifier can be made up of letters, digits, the underscore character (_), and the dollar sign
- Identifiers cannot begin with a digit
- Java is *case sensitive*, therefore `Result` and `result` are different identifiers

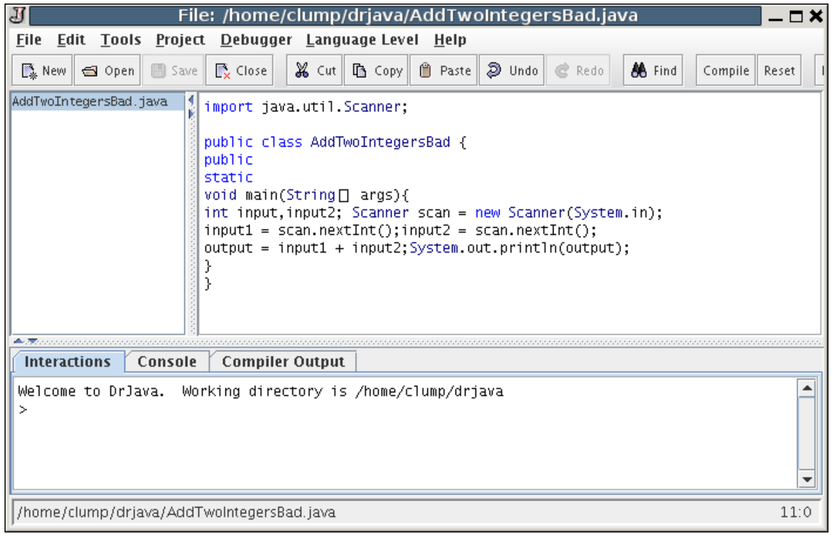
Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as `input1, AddTwoIntegers`)
- Sometimes we are using another programmer's code, so we use the identifiers that they chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
  – A reserved word cannot be used in any other way
  – *Main, class, public, …*

More on *println*

- *println* takes one input
  - a character string: *println("hello world");*
  - the value of a variable: *println(output);*
  - the combination of both:
    - *println("The sum is " + output);*
- We will understand the exact semantics behind this soon

Formatting and Errors


File: /home/clump/drjava/AddTwoIntegersBad.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset
AddTwoIntegersBad.java
Interactions Console Compiler Output

```
import java.util.Scanner;

public class AddTwoIntegersBad {
public
static
void main(String[] args){
int input1,input2; Scanner scan = new Scanner(System.in);
input1 = scan.nextInt();input2 = scan.nextInt();
output = input1 + input2;System.out.println(output);
}
}
```

```
Welcome to DrJava.  Working directory is /home/clump/drjava
>
```

```
/home/clump/drjava/AddTwoIntegersBad.java                11:0
```

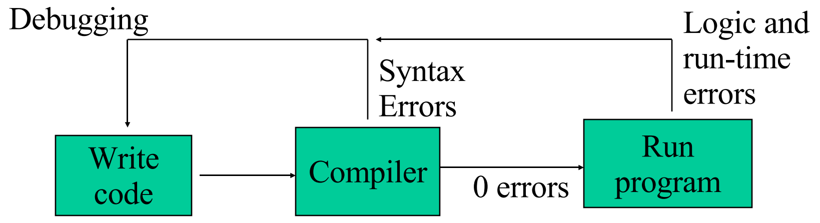
Formatting rules

- Spaces, blank lines, and tabs are collectively called *white space*
  - separates words and symbols in a program
  - Extra white spaces are ignored
- A valid Java program can be formatted many different ways
- Programs should be formatted for readability
  - use proper indentation
  - use space and new lines
  - use comments

Programming Errors

- A program can have three types of errors
  - The compiler will find problems with syntax and other basic issues (*compile-time errors*)
    - If compile-time errors exist, an executable version of the program is not created
  - A program may run, but produce incorrect results (*logical errors*)
    - output = input1 - input2;
  - A problem can occur during program execution, and causes a program to terminate abnormally (*run-time errors*)
    - Divide by zero
    - Wrong data type

Development Life Cycle



Errors may take a long time to debug!

Important Note: When you compile for the first time and see the 150 errors, do not despair. Only the first 1 or 2 errors are relevant. Fix those and compile again. There should be fewer errors (like 50). Repeat until no errors.

Syntax and Semantics

- The *syntax rules* of a language define how we can put symbols, reserved words, and identifiers together to make a valid program (see appendix L)
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we <u>meant</u> to tell it to do

Calculator II: Choosing the right data type

- Integer: -4, -3, -2, -1, 0, 1, 2, 3, …
- Real Number:
  - number that can be given by an infinite decimal representation (e.g, 3.237654…)
- floating point number:
  - approximation of a real number
  - needs only finite space (fits in a cell or set of cells)
  - data type in Java (for now): double

Calculator II: Choosing the right data type

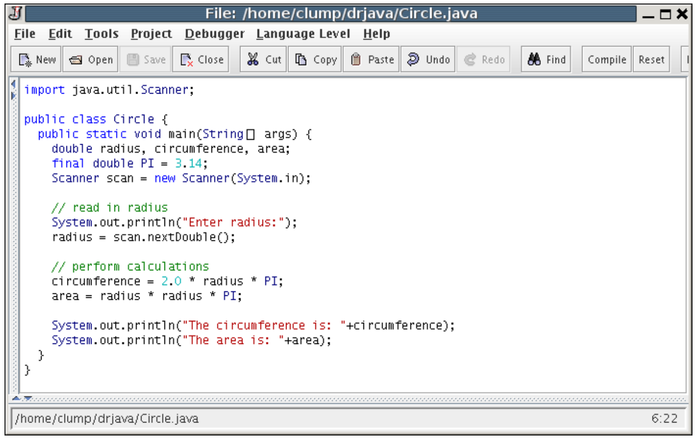
- Integer vs. double

```
import java.util.Scanner;

public class AddTwoDoubles {
  public static void main(String[] args) {
    double input1,input2,output;

    Scanner scan = new Scanner(System.in);

    // input the values
    System.out.println("Add: Type in the first number:");
    input1 = scan.nextDouble();
    System.out.println("Add: Type in the second number:");
    input2 = scan.nextDouble();

    // perform the calculation
    output = input1 + input2;

    System.out.println("The sum is: "+output);
  }
}
```

Data Type compatibility

- If you try to assign a *double* value to a variable of type *int*, you get a run-time error
- If you try to assign an *int* value to a variable of type *double*, an automatic conversion occurs

Calculator III: Constants


File: /home/clump/drjava/Circle.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset

```
import java.util.Scanner;

public class Circle {
  public static void main(String[] args) {
    double radius, circumference, area;
    final double PI = 3.14;
    Scanner scan = new Scanner(System.in);

    // read in radius
    System.out.println("Enter radius:");
    radius = scan.nextDouble();

    // perform calculations
    circumference = 2.0 * radius * PI;
    area = radius * radius * PI;

    System.out.println("The circumference is: "+circumference);
    System.out.println("The area is: "+area);
  }
}
```

/home/clump/drjava/Circle.java 6:22

Constants

- A constant is an identifier that is similar to a variable except that it holds one value for its entire existence
- The compiler will issue an error if you try to change a constant
- In Java, we use the `final` modifier to declare a constant

```
final double PI = 3.14;
```

- Constants:
  - give names to otherwise unclear literal values
  - facilitate changes to the code
    - More precision required: change PI only once to 3.14159
  - prevent inadvertent errors

Arithmetic Expressions

- An *expression* is a combination of operators and operands
  - *`radius * radius * PI`*
- *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

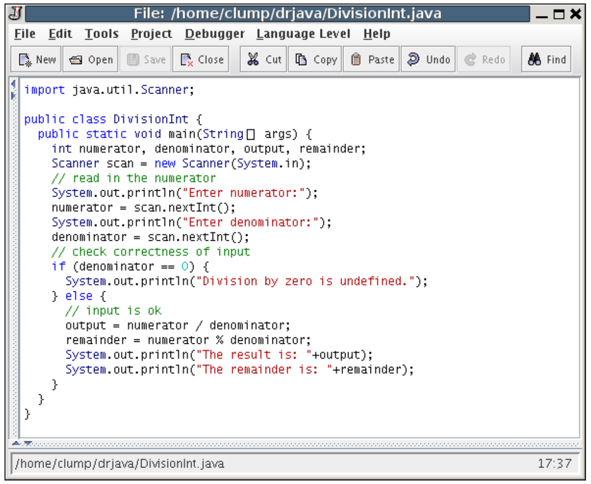
Addition	x + y
Subtraction	x - y
Multiplication	x * y
Division	x / y
Remainder	x % y
Negative	- x

- If either or both operands to an arithmetic operator are floating point (double), the result is floating point (double)

Division with Integers

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)
- The remainder operator (%) returns the remainder after dividing the second operand into the first
- Example 1:
  - int numHours = 52;
  - int fullDays = numHours / 24;
    - 2
  - int remainingHours = numHours % 24;
    - 4
- Division by 0
  - Produces run-time error
  - Program has to avoid it

Calculator IV: Division


File: /home/clump/drjava/DivisionInt.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find

```
import java.util.Scanner;

public class DivisionInt {
  public static void main(String[] args) {
    int numerator, denominator, output, remainder;
    Scanner scan = new Scanner(System.in);
    // read in the numerator
    System.out.println("Enter numerator:");
    numerator = scan.nextInt();
    System.out.println("Enter denominator:");
    denominator = scan.nextInt();
    // check correctness of input
    if (denominator == 0) {
      System.out.println("Division by zero is undefined.");
    } else {
      // input is ok
      output = numerator / denominator;
      remainder = numerator % denominator;
      System.out.println("The result is: "+output);
      System.out.println("The remainder is: "+remainder);
    }
  }
}
```

If-else statements

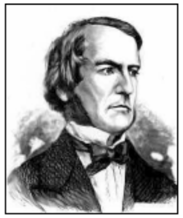
- A statement that allows a program to choose an action depending on the value of a boolean expression
- Example:

```
if (balance > amount)
    Balance = balance – amount;
else
    System.out.println("You cannot withdraw more money
      than you have");
System.out.println("your balance is: " + balance);
```

  - If the value of the variable *balance* is larger than the value of the variable *amount*, the *amount* is subtracted from the *balance*
  - Otherwise the user is informed that the subtraction cannot be done
  - In any case, the value of the *balanc*e is printed

Boolean Expression

- An expression that evaluates either to "true" or to "false"
- Named after George Boole, inventor of the Boolean Algebra (we will discuss it in more detail later)
- Similar concept in natural language
  - "the traffic light is red"
  - This expression is either true or false



Comparison

- Boolean Expressions often contain comparisons;
  - *`if (denominator == 0)`*
    - If the denominator is zero
    - Note the difference of comparison == to assignment =
      - One of the most common errors
  - *`If (denominator != 0)`*
    - If the denominator is not zero
  - *`if (balance > amount), if (balance < amount)`*
    - If the balance is larger / smaller than the amount
  - *`If (balance >= amount)`*
    - If the balance is larger or equal to the amount
  - *`If (balance <= amount)`*
    - If the balance is smaller or equal to the amount

The simple if-then-else Statement

```
if ( condition )
    statement1;
else
    statement2;
```

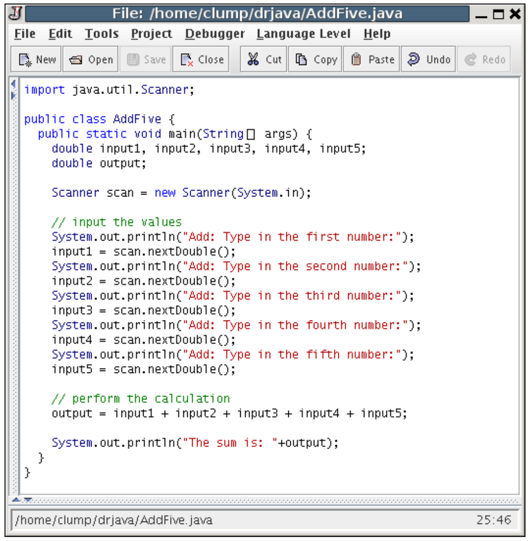
- If the condition is true, statement1 is executed; if the condition is false, statement2 is executed
- One or the other will be executed, not both

Block Statements

- Several statements can be grouped together into a *block statement*
- A block is delimited by braces ({ . . . })
- A block statement can be used wherever a statement is called for in the Java syntax
- For example, in an if-else statement, the if portion, or the else portion, or both, could be block statements

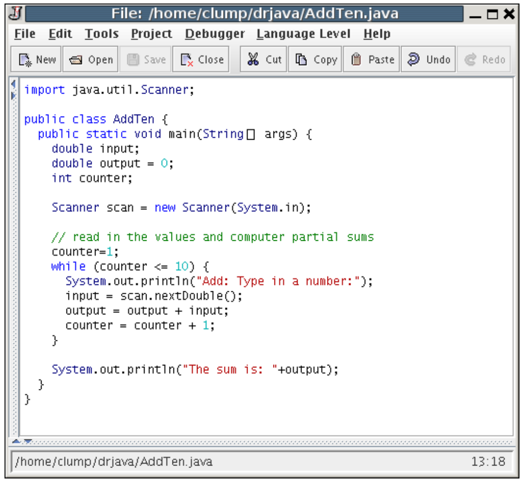
- Task: rewrite the division program with comparison
  - `if (denominator != 0)`

Calculator V: Add five numbers


File: /home/clump/drjava/AddFive.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo
/home/clump/drjava/AddFive.java
25:46

```
import java.util.Scanner;

public class AddFive {
  public static void main(String[] args) {
    double input1, input2, input3, input4, input5;
    double output;

    Scanner scan = new Scanner(System.in);

    // input the values
    System.out.println("Add: Type in the first number:");
    input1 = scan.nextDouble();
    System.out.println("Add: Type in the second number:");
    input2 = scan.nextDouble();
    System.out.println("Add: Type in the third number:");
    input3 = scan.nextDouble();
    System.out.println("Add: Type in the fourth number:");
    input4 = scan.nextDouble();
    System.out.println("Add: Type in the fifth number:");
    input5 = scan.nextDouble();

    // perform the calculation
    output = input1 + input2 + input3 + input4 + input5;

    System.out.println("The sum is: "+output);
  }
}
```

Calculator VI: add 10 numbers


File: /home/clump/drjava/AddTen.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo
/home/clump/drjava/AddTen.java
13:18

```
import java.util.Scanner;

public class AddTen {
  public static void main(String[] args) {
    double input;
    double output = 0;
    int counter;

    Scanner scan = new Scanner(System.in);

    // read in the values and computer partial sums
    counter=1;
    while (counter <= 10) {
      System.out.println("Add: Type in a number:");
      input = scan.nextDouble();
      output = output + input;
      counter = counter + 1;
    }

    System.out.println("The sum is: "+output);
  }
}
```

The while-loop

- A loop allows us to execute a statement or a block of statements repetitively
- Body of the loop: the block of statements contained in the loop (executed repetitively)
- Iteration: one execution of body of the loop
- The body is executed repeatedly as long as the condition after the while evaluates to true
  - If the condition never evaluates to true, then the loop is never executed

The While-loop syntax

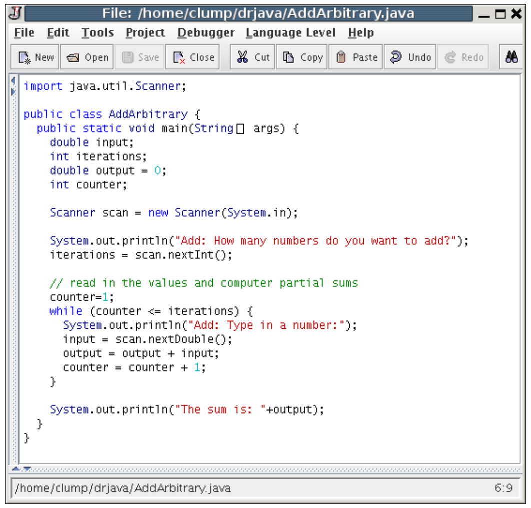
`while` is a reserved word

```
while ( condition )
{
    statement1;
    statement2;
}
```

If the condition is true, the statement is executed. Then the condition is evaluated again.

The statement is executed repetitively until the condition becomes false.

Calculator VII: Adding an arbitrary amount of numbers


File: /home/clump/drjava/AddArbitrary.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo
/home/clump/drjava/AddArbitrary.java
6:9

```
import java.util.Scanner;

public class AddArbitrary {
  public static void main(String[] args) {
    double input;
    int iterations;
    double output = 0;
    int counter;

    Scanner scan = new Scanner(System.in);

    System.out.println("Add: How many numbers do you want to add?");
    iterations = scan.nextInt();

    // read in the values and computer partial sums
    counter=1;
    while (counter <= iterations) {
      System.out.println("Add: Type in a number:");
      input = scan.nextDouble();
      output = output + input;
      counter = counter + 1;
    }

    System.out.println("The sum is: "+output);
  }
}
```

Classes

- So far, we have used some existing classes:
  - Scanner:
    - Allows us to read from keyboard: nextInt, nextDouble, …
  - System.out
    - Allows us to print information to the screen: println
  - We call the operations that we can perform *methods*
- So far, we have developed a set of own classes
  - Division, AddArbitrary, …
  - But are these conceptually classes?
  - They are rather tasks of a class calculator!

The calculator class

- Provides Addition and Division
- Expects repetitive input from user
- User must indicate type of operation (addition, division, …)
- User must indicate input
- Calculator performs operation
- Calculator exits if user does not want to have further computation

Main Method

- Main
  - get type of operation wanted by user
    - 0: exit
    - 1: add
    - 2: divide
  - While (type != 0)
    - If (type == 1)
      - Call Addition method
    - If (type == 2)
      - Call Division method
    - Make sure no other input is provided
    - Get next type of operation wanted by user

Summary

- Variables, variable assignments, expressions are the fundamental building blocks
- Variables can have different data types
  - So far integer and floating point
- We can perform basic operations on variables
  - +, -, *, /
- If-then-else
  - control when certain statements are executed
- While loops
  - execute statements repetitively
- concept of a class
  - bundle related functionality

Problem Solving

- The purpose of writing a program is to solve a problem

- The general steps in problem solving are:
  - Understand the problem
  - Dissect the problem into manageable pieces
  - Design a solution
  - Consider alternatives to the solution and refine it
  - Implement the solution
  - Test the solution and fix any problems that exist

Calculator I: add two Integers

```
import java.util.Scanner;

public class AddTwoIntegers
{
  public static void main (String [] args)
  {
    int input1, input2, output;

    Scanner scan = new Scanner(System.in);

    // read in the values
    System.out.println("Add: Type the first integer:");
    input1 = scan.nextInt();
    System.out.println("Add: Type the second integer:");
    input2 = scan.nextInt();

    // perform calculation
    output = input1 + input2;

    System.out.println("The sum is: " + output);
  }
}
```

Formatting and Errors

```
import java.util.Scanner;

public class AddTwoIntegersBad
{
public
static
void main (String [] args){
int input, input2; Scanner scan = new Scanner(System.in);
input1 = scan.nextInt(); input2 = scan.nextInt()
output = input1 + input2;System.out.println("The sum is: " + output);
}
}
```

Calculator II: Choosing the right data type

- Integer vs. double

```
import java.util.Scanner;
public class AddTwo
{
  public static void main (String [] args)
  {
    double input1, input2, output;
    Scanner scan = new Scanner(System.in);

    // read in the values
    System.out.println("Add: Type the first number:");
    input1 = scan.nextDouble();
    System.out.println("Add: Type the second number:");
    input2 = scan.nextDouble();

    // perform calculation
    output = input1 + input2;
    System.out.println("The sum is: " + output);
  }
}
```

Calculator III: Constants

```
import java.util.Scanner;
public class Circle
{
  public static void main (String [] args)
  {
    double radius, circumference, area;
    final double PI = 3.14;
    Scanner scan = new Scanner(System.in);

    // read in the radius
    System.out.println(Enter radius:");
    input1 = scan.nextDouble();

    // perform calculation
    circumference = 2 * radius * PI;
    area = radius * radius * PI;
    System.out.println("The circumference is: " + circumference);
    System.out.println("The area is: " + area);
  }
}
```

Calculator IV: Division

```
import java.util.Scanner;
public class DivisionInt
{
  public static void main (String [] args)
  {
    int nominator, denominator, output, remainder;
     Scanner scan = new Scanner(System.in);
    // read in the input
    System.out.println("Enter nominator:");
    nominator = scan.nextInt();
     System.out.println("Enter denominator:");
    denominator = scan.nextInt();
     // check correctness of input
    if (denominator == 0)
           System.out.println("The denominator may not be 0");
    else // perform calculation
    {
           output = nominator / denominator;
           remainder = nominator % denominator;
           System.out.println("The result is: " + output);
           System.out.println("The remainder is: " + remainder);
    }
  }
}
```

Calculator V: Add five numbers

```
import java.util.Scanner;
public class AddFive
{
  public static void main (String [] args)
  {
    double input1, input2, input3, input4, input5, output;
     Scanner scan = new Scanner(System.in);
    // read in the input
    System.out.println("Enter first number:");
    input1 = scan.nextDouble();
    System.out.println("Enter second number:");
    input2 = scan.nextDouble();
    System.out.println("Enter third number:");
    input3 = scan.nextDouble();
    System.out.println("Enter fourth number:");
    input4 = scan.nextDouble();
    System.out.println("Enter fifth number:");
    input5 = scan.nextDouble();

     // perform calculation
    output = input1 + input2 + input3 + input4 + input5;
    System.out.println("The result is: " + output);
  }
}
```

Calculator VI: add 10 numbers

```
import java.util.Scanner;
public class AddTen
{
  public static void main (String [] args)
  {
    double input;
    double output = 0;
    int counter;

    Scanner scan = new Scanner(System.in);

    // read in the values in a loop and incrementally perform calculation
    counter = 1;
    while (counter <= 10)
    {
      System.out.println("Enter number:");
      input = scan.nextDouble();
      output = output + input;
      counter = counter + 1;
    }
    System.out.println("The sum is: " + output);
  }
}
```

Calculator VII: Adding an arbitrary amount of numbers

```
import java.util.Scanner;
public class AddArbitrary
{
  public static void main (String [] args)
  {
    double input;
    int iterations;
    double output = 0;
    int counter;

    Scanner scan = new Scanner(System.in);

    System.out.println("Indicate the amount of number:");
    iterations = scan.nextInt();
    // read in the values in a loop and incrementally perform calculation
    counter = 1;
    while (counter <= iterations)
    {
      System.out.println("Enter number:");
      input = scan.nextDouble();
      output = output + input;
      counter = counter + 1;
    }
    System.out.println("The sum is: " + output);
  }
}
```