 **File**: save as a Java file with the same name as the public class name.

**CSD**: finds big errors, like unmatched parenthesis, braces and brackets. Aligns your code to make it easy to read.

**Compile**: finds small errors, like missing semicolon, spelling and case-sensitivity issues.

**Run**: executes last successful compile.

**Debug**:if your program compiles but does not execute correctly, you can run the program one

line at a time and watch the variables change with the debugger. Set up a break point

at the part of the program you want to debug, hit the Debug button and step through

the code to find the logical error.

**Break Point:** along the left border of the code window, toggle red-dot for a break point where

you want to start debugging.

**Step:** when debugging, step will execute a single line of code

**Step into:** when debugging, step into will enter a method call to allow you to debug within.

**End:** halt a program if you have an infinite loop.

**import java.io.\*; //INFO**

**import java.util.\*;  
public class HowToGetInput**  
{  
 //any method that gets input from a file or keyboard should throw an IOException  
 **public static void main(String argv[]) throws IOException**   
 {  
 **String sentence;** //a collection of characters  
 **int num;** //a whole number  
 **double real;** //a number with a decimal  
 **Scanner input = new Scanner(System.in);**  
  
 //TO ENTER A WORD OR WORDS  
 **System.out.println("Enter a sentence");**  
 **sentence = input.nextLine();**   
 //waits for input and returns the sentence that was typed as a String  
  
 //TO READ AN INTEGER   
 **System.out.println("Enter a whole number");**  
 **num = input.nextInt();**   
 //waits for input and returns what was typed as an int  
  
 //TO READ A DOUBLE  
 **System.out.println("Enter a real number");  
 real = input.nextDouble();**   
 //waits for input and returns what was typed as a double  
  
 **System.out.println(sentence + " " + num + " " + real);** /\*separate writing variables and text with the + operator  
 anything within quotes is just writing text  
 anything not in quotes must be a variable or object name \*/   
 }  
}

EXPLANATIONS:

Any text after // or in between /\* and \*/ will be a comment and not affect the code.

The public class HowToGetInput must be in a program called HowToGetInput.java

A String type is a collection of characters (word or sentence). An int is a whole number (no decimal allowed) and a double is a number that can have a decimal.

System.out.println(); is a method call that sends information out to the screen.

The assignment operator (=) will assign a value to the state of a variable (int/double) or an object (String).

**Scanner input = new Scanner(System.in);**

The Scanner object allows us to read in values form a file or typed in from the keyboard by the person who runs the program.

Scanner methods include:

next(); reads in text up to the first space or enter key and

returns it as a String

nextLine(); reads in text up to the first enter key and returns it

as a String

nextInt(); reads in a whole number and returns it as an int

nextDouble(); reads in a number (decimals are ok) and returns it as a

double

**System.out.println("The variable x stores the value " + x);**

The System object will allow us to write text and the values of variables and objects out to the screen. Separate text and variables/objects using the + operator.

**INFO**

Here is a simple method that will ask for a radius of a circle, and then find and show you the area and perimeter (circumference) of the circle.

**//EXAMPLE: CALCULATE THE AREA AND PERIMETER OF A CIRCLE GIVEN THE RADIUS**

**public class Circle**

**{  
 public static void main(String[]arg) throws IOException  
 {**

**Scanner input = new Scanner(System.in);**

**//FIRST, DEFINE THE VARIABLES THAT YOU WILL NEED. GIVE THEM NAMES THAT MAKE SENSE  
 double radius, //user inputted radius of a circle  
 area, //the area of a circle  
 perimeter; //perimeter of a circle**

**//INPUT - ASK THE USER FOR DATA AND STORE IT INTO A VARIABLE  
 System.out.println("Enter the radius of a circle:");  
 radius = input.nextDouble();**

**//PROCESS - HAVE THE COMPUTER DO THE WORK FOR YOU  
 area = Math.PI \* radius \* radius;  
 perimeter = 2 \* Math.PI \* radius;**

**//OUTPUT - SHOW THE USER THE RESULTS OF THAT WORK  
 System.out.println("The area of a circle with radius "+radius+" is "+area+".");   
 System.out.println("The perimeter is " + perimeter + ".");  
 }  
}**

Items to note:

\* **Step 1** is to **define any variables that you need**. If when given a radius of a circle, you need to have the computer find the area and perimeter of a geometric shape, then you need to define three variables. Give the variables names that make sense, instead of vague names like x, y or z.

\* Variable Scope: variables and objects only exist within the { } that they are defined in. When the method is over, the variables radius, area and perimeter no longer exist. If you were to define a variable or object inside of an if-statement or a loop, it would no longer exist outside of the if-statement or loop.

\* **Step 2** is to **ask the user for input** of some kind. Any method that gets input from the keyboard or from a file will have to throw an IOException. This prepares the method for knowing how to handle bogus input (trying to read in a number and getting a word instead). You must also import java.io.\*;

\* **Step 3** is to **process the input and find the answer**. Remember that with an assignment statement, the variable on the left is the one that changes value.

\* **Step 4** is to **show the results to the user**. In a System.out.println statement, you can go back and forth between writing text and the state of variables by separating them with the + operator. Anything inbetween “” will be text, written verbatim out to the screen. Any variable name will write the state of the variable out to the screen.

Example: if a variable called x stores the value 3, then

System.out.println*(“*Your number is *“* + x + *“* and its square is *“* + (x\*x) + *“*!*”*);

would write: Your number is 3 and its square is 9!

APCS Exercises Math Operators Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

Java math operators include: + (addition) - (subtraction)

\* (multiplication)

/ (division, given at least one double)

example: 3.0 / 6.0 returns the expected value of 0.5.

/ (div, or whole number division, given two ints)

example: 3 / 6 returns 0 because 6 “goes into” 3 zero times.

% (modulous, or mod – the remainder from div)

example: 9 % 4 retuns 1 because 4 “goes into” 9 two times with a remainder of 1.

Math.sqrt(x) (square root, returns a double)

Math.sin(x) Math.cos(x) Math.tan(x)

Math.abs(x) (the absolute value of x)

A variable as an int or double can change type via “casting”, like this:

double x = 3.9;

System.out.println((int)(x)); //writes 3

Standard order of operations are maintained.

Example: Show the result outputted to the screen:

double ans = ((Math.sqrt(9 % 5) + (1.5));

System.out.println((int)(ans));

ans will store the value 3.5 because 9%5 yields 4, and the square

root of 4 added to 1.5 is 3.5.

Solution: 3 (because the integer version of 3.5 is 3)

Show the result outputted to the screen:

1. System.out.println ( 9 / 5 + 5 / 9);

2) int ans = 5 % 12;

System.out.println(ans / 3.0);

1. System.out.println(Math.abs((3 – 4) \* Math.sqrt(81 / 9)));

4) int ans = ((int)(Math.sqrt(144))) % 10;

System.out.println(ans);

1. double x = 4.0 / 8.0;

int y = 4 / 8;

double ans = x + y;

System.out.println(ans);

1. int num = 1234;

int w = num % 10;

int x = num / 10;

int y = num % 100;

int z = num / 100;

System.out.println(w + “ “ + x + “ “ + y + “ “ + z);

You can update the value of a variable by referring to its previous value:

int x = 47;

x = x + 10; //reads x becomes its old value plus 10.

//x now holds the value 57

The operator ++ will add one to the previous value of the variable.

The operator – will subtract one from the previous value of the variable.

So the statement x++; is a shortcut for the assignment statement x = x + 1;

In an assignment statement, only the variable or object to the left of the assignment operator (=) gets changed to a different value.

i.e., to say x = y + 10; means that x will change to store y’s value + 10, but y doesn’t change at all.

Consider each block of code, and write the state of each variable after the code is executed.

7) int a = 3;

int b = 5;

b++;

a = a + 10;

a = b; a\_\_\_\_\_ b\_\_\_\_\_

8) int x = 10;

int y = -3;

x = x + y;

y = x; x\_\_\_\_\_ y\_\_\_\_\_

9) double n = 3.5;

double m = n;

n--;

m++; n\_\_\_\_\_ m\_\_\_\_\_

10) double p = 2.0;

double q = 6.0;

p = p \* q;

q = q + p;

p++; p\_\_\_\_\_ q\_\_\_\_\_

APCS Exercises Div and Mod Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

1) Consider you want to have the following program:

VOLLEYBALL TEAMS ARE SET TO 7 PLAYERS PER TEAM. HAVE THE USER ENTER THE

NUMBER OF AVAILABLE PLAYERS AND THEN CALCULATE THE NUMBER OF TEAMS AND

PLAYERS LEFT OVER.

Ex. Enter the number of players: 37

There will be 5 teams with 2 left over.

Complete the code so that it works as specified:

int numPlayers, numTeams, numLeft;  
   
 System.out.println("Enter the number of players:");  
 numPlayers = input.nextInt();

//complete the statements here  
 numTeams =

numLeft =

System.out.println("There will be “+numTeams+” teams

with “+numLeft+” left over.");

2) Consider you want to have the following program:

THE USER ENTERS A TIME IN MINUTES AND THE METHOD CALCULATES AND SHOWS THE

NUMBER OF HOURS AND MINUTES.

Ex. Enter the number of minutes: 327

This is 5:27

Write the code to work as specified (use number 1 as a model)

3) Consider you want to have the following program:

THE USER ENTERS AN AMOUNT OF CHANGE IN CENTS AND THE PROGRAM SHOWS THE NUMBER

OF QUARTERS, DIMES, NICKLES AND PENNIES.

Ex. Enter an amount of change in cents: 68

Quarters: 2 Dimes: 1 Nickles: 1 Pennies: 3

Write the code to work as specified (use number 1 as a model)

4) Develop your own program with the following criteria:

User inputs into int variables

Solution(s) are found using div and mod

Must be different from the programs on the first side.

**import java.io.\*;**

**import java.util.\*;  
public class DivAndMod4**  
{  
 **public static void main(String argv[]) throws IOException**   
 {

}  
}

**Exercises Casting and Order of Operations** Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

Examples: You can cast back and forth between **double**s and **int**s.

(**int**)(9.18) knocks off the decimal resulting in 9.

(**double**)(3) returns a real number, resulting in 3.0

Write the resulting returned value or show the resulting output (whichever applies):

1) (**int**)(-4.53); \_\_\_\_\_\_\_\_\_\_

2) **double** x = 12.999999;

System.out.println((**int**)(x) -1); \_\_\_\_\_\_\_\_\_\_

3) (**double**)((**int**)(9.0 / 4.0)) \_\_\_\_\_\_\_\_\_\_

4) **int** x = 15;

System.out.println((**int**)(0.94) + (**double**)(x)); \_\_\_\_\_\_\_\_\_\_

5) (**int**)(Math.PI) \_\_\_\_\_\_\_\_\_\_

6) (**double**)(3 + 5) \_\_\_\_\_\_\_\_\_\_

7) Consider the following incomplete code. Complete the assignment statement so that the solution displayed is the result of x divided by y, as opposed to whole number division. If the user enters 5 for x and 2 for y, you want the output to display The quotient is 2.5

int x, y, ans;

System.out.println(“Enter a number”);

x = input.nextInt();

System.out.println(“Enter a number”);

y = input.nextInt();

System.out.println(“The quotient is “ +

Consider we have the following variables:

int x = 4;

int y = 5;

double z = 0.5

String word = “cheese”;

Write the resulting value of the following statements:

8) x + y + word \_\_\_\_\_\_\_\_\_\_

9) x + word + y \_\_\_\_\_\_\_\_\_\_

10) word + x + y \_\_\_\_\_\_\_\_\_\_

11) x \* z; \_\_\_\_\_\_\_\_\_\_

12) x / y + word \_\_\_\_\_\_\_\_\_\_

13) word + x / y \_\_\_\_\_\_\_\_\_\_

14) word + (x + y) \_\_\_\_\_\_\_\_\_\_

15) (int)(y \* z) \_\_\_\_\_\_\_\_\_\_