**Chapter 2 – Elementary Programming**

|  |  |  |  |
| --- | --- | --- | --- |
| Sections | Pages | Review Questions | Programming Exercises |
| 2.1-2.16 | 26-59 | 1-11, 13-17, 20-22, 25, 27, 30-32 | 2,4,6,12,14,16,18 |

**Terminology**

algorithm, identifier, variable & naming convention, statement, expression, declaring variables, constants & naming convention, numeric data types, numeric operators, remainder operator, unary & binary operators, numeric literals, order of operations, integer division, shorthand operators, numeric type conversions, type casting, widening/narrowing a type, Unicode & ASCII code, char data type, string data type, string concatenation, console input, Scanner class, Math class, java.lang package, import statement, System class, block styles, indention, syntax error, runtime error, logic error, debugging, JOptionPane class, showInputDialog method, Integer.parseInt, Double.parseDouble

**Software Lifecylce**

The *software lifecycle* refers to how we go about developing a software system. In our class, we will narrow this down to think about a *program* as opposed to a software system. A very simple software lifecycle is performing these steps:

1. **Analysis** – figuring out *what* the problem is. In our course, this will entail reading a problem from the text or a handout, understanding it, and asking any questions you have about it. Many times this involves use paper and pencil to work out by hand what certain computations are supposed to do. In real world situations, analysis involves talking to and observing people who will use the system.
2. **Design** – figuring out *how* to solve the problem. In our course, this involves figuring out an *algorithm* to solve the problem. In the real world, design also is a large process involving architecture design, database design, network design, security design, user-interface design, and detailed design (the only part we are concerned with in this course).
3. **Coding** – writing the code.
4. **Testing** – designing ways to test your code. This involves specifying the input for a program and making sure the code produces the same results as worked out by hand. We are trying to *break* our code. When we do, we must go back to analysis or design.

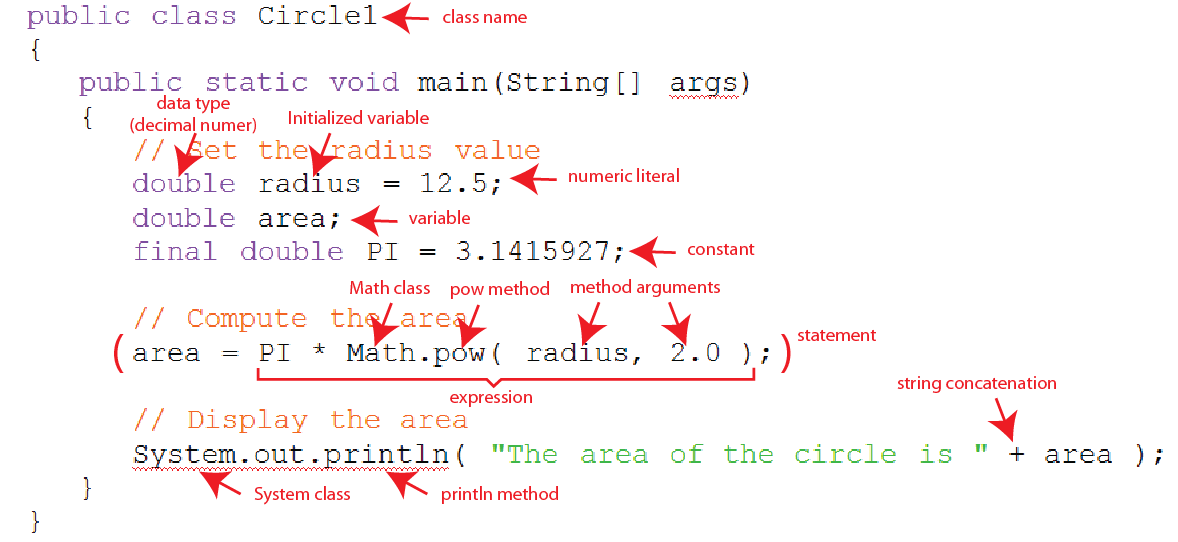
**Example 1**

Problem: Compute and display the area of a circle with radius 12.5 inches.

1. Algorithm – a set of steps which leads to the correct solution to a problem
2. Set the radius to 12.5
3. Compute the area
4. Display the area
5. Program

public class Circle1   
{  
 public static void main(String[] args)   
 {  
 // Set the radius value  
 double radius = 12.5;  
 double area;  
 final double PI = 3.1415927;  
   
 // Compute the area  
 area = PI \* Math.pow( radius, 2.0 );  
   
 // Display the area  
 System.out.println( "The area of the circle is " + area );  
 }  
}

1. Terminology



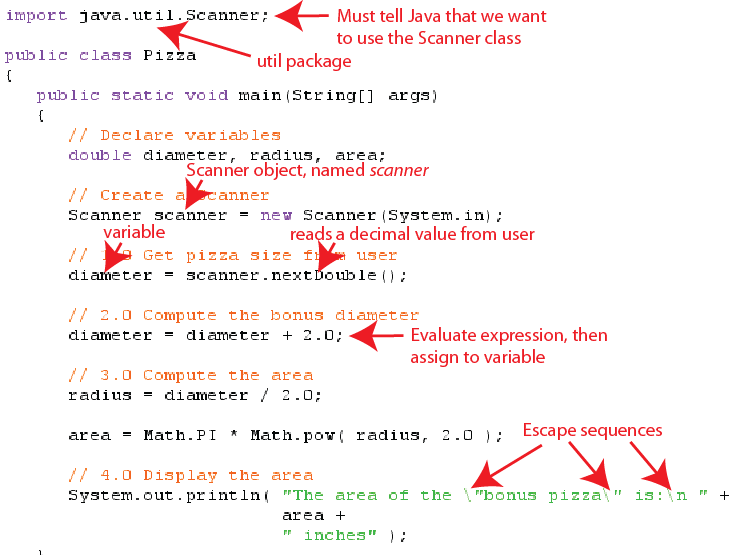
**Example 2**

Problem: A pizza store has a special going on. The customer specifies the size of pizza they want (the diameter in inches) and the pizza store automatically makes them a pizza that is 2 inches wider. Prompt the user for the size of pizza they want. Then, compute and display the area.

1. Algorithm
2. Get the diameter from user
3. Compute the bonus diameter
4. Compute the area
5. Display the area
6. Program – Run in class

import java.util.Scanner;  
  
public class Pizza   
{  
 public static void main(String[] args)   
 {  
 // Declare variables  
 double diameter, radius, area;  
   
 // Create a Scanner  
 Scanner scanner = new Scanner(System.in);  
  
 // 1.0 Get pizza size from user  
 diameter = scanner.nextDouble();  
   
 // 2.0 Compute the bonus diameter  
 diameter = diameter + 2.0;  
   
 // 3.0 Compute the area  
 radius = diameter / 2.0;  
   
 area = Math.PI \* Math.pow( radius, 2.0 );  
   
 // 4.0 Display the area  
 System.out.println( "The area of the \"bonus pizza\" is:\n " +   
 area +  
 " inches" );  
 }  
}

1. Terminology



The results of running the program show:

The area of the "bonus pizza" is:  
7.0685834705770345 inches

**Integer Division and Remainder**

1. Integer division and the remainder operator
2. *int* is another numeric data type that is used to represent an integer. For example,

int x = 4;

1. When you divide two integers, and store the result in an integer, then you lose (truncate) the fractional part. For example:

int x = 14, y = 5, z;

z = x / y; // result is 2

z = 3 / 4; // result is 0

This can be very useful as we will see shortly.

1. % is called the *remainder* operator. When two integers are divided it provides the (integer) remainder. For example:

int x = 14, y = 5, z;

z = x % y; // result is 4

z = 3 % 4; // result is 3

This, also can be very useful as we will see next.

1. Problem: Convert the value 336 seconds into minutes and seconds and display.
2. Algorithm
3. Set the value of total seconds
4. Compute the number of minutes
5. Compute the number of seconds
6. Display the minutes and seconds
7. Program

public class Minutes   
{  
 public static void main(String[] args)   
 {  
 // Declare variables  
 int minutes, seconds;  
 String message;  
   
 // 1.0 Set the total number of seconds  
 int totalSeconds = 336;  
   
 // 2.0 Compute the number of minutes  
 minutes = totalSeconds / 60;  
   
 // 3.0 Compute the number of seconds  
 seconds = totalSeconds % 60;  
   
 // 4.0 Display to results  
 message = totalSeconds + " seconds = " +  
 minutes + " minutes and " +  
 seconds + " seconds";  
   
 System.out.println( message );  
 }  
}

1. Problem: Convert the value 9214 seconds into hours, minutes and seconds and display.
2. Analysis & Design. *Analysis* involving figuring out exactly what the problem is, understanding the problem, asking for clarification if necessary. *Design* is figuring out a solution to the problem.
3. 9214 seconds = how many minutes and seconds?

9214 *sec* / 60 *sec/min* = 153 *min*

9214 % 60 = 34 *sec.*

Thus, 9214 *sec* = 153 *min* and 34 *sec*

1. 153 minutes = how many hours and minutes?

153 *min* / 60 *min/hr* = 2 *hr*

153 % 60 = 33 *min*

Thus, 153 *min*  = 2 *hr* and 33 *min*

Thus, 9214 *sec* = 2 *hr* 33 *min* and 34 *sec*

Note that we will need a variable to represent the *total minutes* (153) and the *minutes* (33)

1. Algorithm
2. Set the value of total seconds
3. Convert total seconds to total minutes and seconds
4. Convert total minutes to hours and minutes
5. Display the hours, minutes and seconds
6. Program

public class Hours   
{  
 public static void main(String[] args)   
 {  
 // Declare variables  
 int minutes, seconds, hours, totalMinutes;  
 String message;  
   
 // 1.0 Set the total number of seconds  
 int totalSeconds = 9214;  
   
 // 2.0 Convert total seconds to total minutes and seconds  
 totalMinutes = totalSeconds / 60;  
 seconds = totalSeconds % 60;  
  
 // 3.0 Convert total minutes to hours and minutes   
 hours = totalMinutes / 60;  
 minutes = totalMinutes % 60;  
   
 // 4.0 Display the results  
 message = totalSeconds + " seconds = " +  
 hours + " hours " +  
 minutes + " minutes and " +  
 seconds + " seconds";  
   
 System.out.println( message );  
 }  
}

**GUI Input**

1. The JOptionPane class has two important (static) methods. We have already seen that the showMessageDialog() method is used to display a message to the user. The showInputDialog() method is use to read information from the user. This method always returns a String. If the information supplied is numeric, we must convert the string number into a proper number. For instance,

String strNumPizzas;  
int numPizzas;  
   
strNumPizzas = JOptionPane.showInputDialog(

"How many pizza's do you want?");

numPizzas = Integer.parseInt( strNumPizzas );

Usually, we would write this code more succinctly:

numPizzas = Integer.parseInt(

JOptionPane.showInputDialog("How many pizza's do you want?") );

1. Problem: Prompt the user for her name and the number of pizzas she wants. Pizza’s cost $12.99 each and tax is 7%. Print an informative receipt to the console that looks something like this:

Receipt

--------------------------------------------------

Customer: Linda Staples

Number of Pizza’s: 3

Cost: $38.97

Tax: $2.7279

Total Cost: $41.6979

1. Algorithm
2. Read name and number of pizzas
3. Compute order costs
4. Build and display receipt
5. Program

import javax.swing.JOptionPane;  
  
public class PizzaGUI   
{  
 public static void main(String[] args)   
 {  
 // Declare variables  
 String namePrompt = "What is your name?";  
 String pizzaPrompt = "How many pizza's do you want?";  
 String receipt, name;  
 int numPizzas;  
 double pizzaCost=12.99, taxRate = 0.07, orderCost,

taxCost, totalCost;

// 1.0 Read name and number of pizzas  
 name = JOptionPane.showInputDialog( namePrompt );  
 numPizzas = Integer.parseInt( JOptionPane.showInputDialog(

pizzaPrompt ) );

// 2.0 Compute order costs  
 orderCost = numPizzas \* pizzaCost;  
 taxCost = orderCost \* taxRate;  
 totalCost = orderCost + taxCost;  
   
 // 3.0 Build and display receipt  
 receipt = "Receipt\n";  
 receipt += "--------------------------------\n";  
 receipt += "Customer :\t" + name + "\n";  
 receipt += "Num Pizzas:\t" + numPizzas + "\n";  
 receipt += "Cost :\t" + orderCost + "\n";  
 receipt += "Tax :\t" + taxCost + "\n";  
 receipt += "Total Cost:\t" + totalCost;  
   
 System.out.println( receipt );  
 }  
}

**Powerpoint**

Cover PowerPoint slides for Chapter 2 in class.