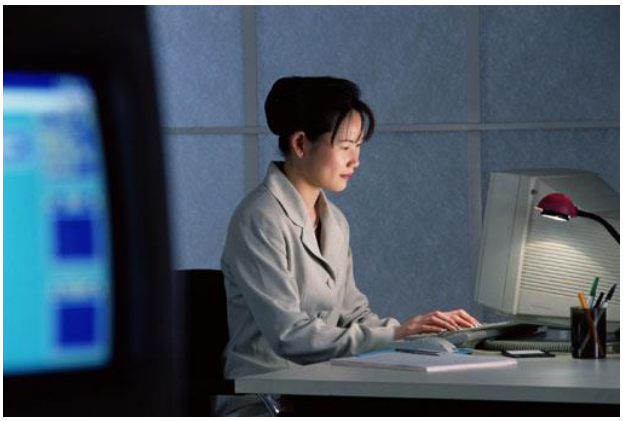
Programming Exercise 3:
User Input



Purpose: How to get information from the user. Converting from one type to another.

Background readings from textbook: Liang,
Programming Style: 1.9
Casting: 2.15 (also 4.4.10)
Software Development Process: 2.16
Reading input from the Console: 2.3

Due date for section 001: Monday, February 1 by 10 am
Due date for section 002: Wednesday, February 3 by 10 am

Overview

Most applications will require input either from the user (via keyboard) or from stored data on files. In programs 1 and 2, we "hard coded" the input as literal values but this does not provide the program with any flexibility. Instead, we need to know how to obtain user input. In Java, user input is handled through objects (instances of classes). It used to be more of a hassle in that you had to set up all kinds of things to do input and then input was only acceptable as Strings so that if you wanted to treat input as a number or character or other, you had to convert it with a type of casting statement. The most recent versions of Java have included a new class called the Scanner. With the Scanner, things are far easier. In this assignment, we will use the Scanner to get input and use various forms of conversions to change data types including casts like we did in lab 2 but also using other approaches.

Part 1: Java Examples

1. Request information from the user.

```
Scanner input = new Scanner (System.in);
System.out.println("Enter your name.");
String fullName = input.next();
```

New Scanner() creates an object of the Scanner type. System.in makes the program obtain the input from the console (keyboard). Scanner represents a stream of data for input.

To use the **Scanner** class import the package **java.util.Scanner** at the top of your program. Notice how we precede the actual input (input.next();) with an output statement. The output statement is called a *prompting message*, or a *prompt* for short. Our prompt uses a println statement. This will cause the user's typed input to appear on a separate line. If we want to have the user's input appear on the same line as the prompt, we use a print statement instead. We might want to convert our prompt to be `System.out.print("Enter your name  ");` The added spaces make

sure that there is some blank space between our message and their input. With the use of println, we get the following:

```
Enter your name
Richard
```

Using a print statement with no blank spaces after name, we get the following:

```
Enter your nameRichard
```

Using the above print statement, we get the following:

```
Enter your name   Richard
```

It's a stylistic choice about whether to use print or println, but if you use print, make sure you have blank spaces between the prompt and the input.

2. Request double data from the user and convert what they type to a **double** value.

```
System.out.println("Enter the price per gallon.");
String pricePerGallonStr = input.next();
double gallonPrice = Double.parseDouble(pricePerGallonStr);
```

The statement `input.next( );` tells the input variable (which is an instance of a Scanner) to execute its next method. The next method obtains whatever String was input and returns it. Here, we want to store the result as a double, not as a String. So we are required to convert it. We convert Strings to other types using the notation shown above (`Double.parseDouble` for a double, `Int.parseInt` for an int). However, the Scanner class can accept non-String types. We can use `input.nextInt( );` to get an int and `input.nextDouble( );` to get a double. Our revised code looks like the following.

```
System.out.println("Enter the price per gallon." );
double gallonPrice = input.nextDouble();
```

3. In the last lab, we used an explicit cast to convert an int value to a double. We can also work things the other way. Here, we are explicitly converting a double to an int. The cast simply chops off any fractional portion, so 33125.58 becomes 33125.

```
double salary = 33125.58;
int dollars = (int) salary;
```

We can *round* a floating point value to the nearest integer by being a little more clever.

```
double salary = 33125.58;
int dollars = (int) (salary + 0.5);
```

We can also accomplish this using the round method through the Math class by doing `Math.round(salary);` (the Math class is covered in section 4.2 if you want to explore it).

4. Use close to close the input stream in order to indicate that there is no more input.

```
input.close();
```

Part 2: Common Pitfalls

1.
```
System.out.println("What is your name? ");
input.next();
```

Logic error!

You must assign the value resulting from input.next( ); to a variable otherwise the input value cannot be used!

2.
```
double weight = 14.3;
int ounces = weight * 16;
```

Syntax error!

*You cannot assign a **double** value to an **int** variable without an explicit cast.*

3.
```
String numberStr = "153";
int value = 14 * numberStr;
```

Syntax error!

A number represented by a string is not numeric; convert it to a numeric data type before using arithmetic on it.

Part 3: A Simple Program

What follows is a simple program that obtains different types of input, uses some of the input for a computation and then outputs the results. Create a new project called Program3a, enter this program, save, compile and debug it (if you have syntax errors). This is practice for the program you will do as your assignment below (Parts 4-6).

```
import java.util.Scanner;

public class Program3a {
	public static void main(String[] args) {
		Scanner in=new Scanner(System.in);
		String city1, city2;
		int distance;
		double costPerMile;
		int dollarAmount;
		System.out.print("Enter the starting city: ");
		city1=in.next();
		System.out.print("Enter the ending city: ");
		city2=in.next();
		System.out.print("Enter distance in miles between cities:  ");
		distance=in.nextInt();
		System.out.print("Enter the cost of travel per mile:  ");
		costPerMile=in.nextDouble();
		dollarAmount=(int)(distance*costPerMile);
		System.out.println("\n\nThe cost of traveling from " +
			city1 + " to " + city2 + " is $" + dollarAmount);
	}
}
```

Run the program inputting Cincinnati, Columbus, 115 and .065. Your output should be $7.

Part 4: Problem

You will write a similar program to the above, but call it Program3b. Here, you will compute a car's gas mileage and the amount you spend per mile traveled. In particular, ask the user for the following.

- their first name (or their full name) (String),

- the price per gallon of gasoline paid at the time of the last fill-up (double),
- the total amount paid for the fill-up (double),
- an initial odometer reading (int),
- a final odometer reading (int).

After inputting these values, compute the total miles drive, the number of gallons of gas used (total amount for fill-up / price per gallon), and the miles per gallon that the car achieved (total miles / number of gallons). Note that number of gallons and miles per gallon will both be doubles. Print this information using System.out.println statements. Below is a sample of the input and the output when running this program.

```
Enter your name: Richard
Enter price per gallon: 1.99
Enter total price for fill up: 12.84
Enter initial odometer value: 51384
Enter final odometer value: 51577


Richard, you drove 193 miles using
6.452261306532663 gallons with an mpg of 29.911993769470406
```

Write this program (make sure you comment the code as you write it). When you have your program written, save and compile it. If it does not compile, fix your syntax errors. Once it does compile, run it on the above input to see if you get the same output. If you get incorrect output, look through your code and try to fix whatever logical errors you might have.

The output is not particularly readable. Let's force the output to look nicer in two ways. First, we will output the number of gallons used as an int. Do this by changing your numberOfGallons variable (whatever you called it) to an int and casting the value you compute for this variable to an int, similar to how we did this with dollarAmount in the previous program. Second, import DecimalFormat (from java.text), create a variable of type DecimalFormat called df with a format of ##.# and then change the System.out.println statement to format the mpg value using your df.format. Refer back to program 2 if you are unsure how to do this.

Part 5: Test Your Program and Submit it

Run your program 4 times, once each on the following data. Collect all of the input and output (copy and paste it) and either paste it into a separate text file, or as a comment at the bottom of your source code. Print out the source code and input/output and hand it in to your instructor or email it (as one or two files depending on whether you save the input/output to a separate file) to your instructor. NOTE: do not submit Program3a.

Name	Price per gallon	Price for fill up	Initial odometer	Final odometer
Your name	2.84	20.25	6144	6279
Frank	1.72	31.40	21975	22496
Ruth	3.75	22.87	89108	89183
Gail	2.05	20.50	65380	65691