Title: Input (using Scanner)

Author: Jonathan Huisingh

Assignment Definition And General Feedback By Michael Panitz at Cascadia Community College (http://www.cascadia.edu)

Table of contents:

(Required sections have a * after them in the above list)

Quick Summary:

In Java, you can receive user input via a Scanner (we will be using a keyboard in this section). The user input will be processed via jGRASP's console window, and you must start the program before it will be accepted. While the program is being run, it will pause when you call for user input. The input will then be processed based on how you programmed the input. You will need to specify what type of input the user should use (int, char, string...).

In order to process user information you will need to import the java util library, in addition to importing the library, you will need to call the Scanner in any class you choose to process user information. Once per class is sufficient, java will allow you to call the Scanner multiple times, but there is no reason to do so.

It is possible to use the keyboard to input, and Java to process all sorts of information, however, we will be focusing only on numerical input (specifically integer) in this section.

When To Use This / Avoid This:

Calling for user input would be prudent whenever you want the user to make a decision as to how the program should proceed. For instance, if you want the user to decide what direction your robot should travel, how far the robot should travel, whether or not to pick up a thing at an intersection, or if you would like the robot to put down a thing at an intersection.

You should not call for user input if there is no reason to have user input. For example, if your robot is designed to simply traverse terrain or a hallway, there would be no point of asking the user how to achieve this.

Example Of Usage:

Lets say you are creating a robot that will have a choice take a left path and pick

up a thing, or take a right path and put a thing down. You can ask the user to decide for you, once you have programmed in both scenarios. This program demonstrates one way to do this, with the new/important parts in **bold-faced larger font**.

There are 5 steps:

- Import the java util file to be able to process the keyboard input.
- Add the line: "Scanner keyboard = new Scanner(System.in)" to your class or method; this is necessary for java be able to use the input device.
- Use System.out.print("Insert text here"); to print out a message telling the user the valid options.
- Call keyboard input with the command keyboard.hasNextInt(); to prompt the user with options for valid input choices.
- Handle the users input correctly based on their choice.

Line	Code
1	`import becker.robots.*;`
2.	`import java.util.*;`
3.	
4.	`class RobotIO extends RobotSE{`
5.	`public RobotIO(City c, int st, int ave, Direction dir, int num){`
6.	`super(c, st, ave, dir, num);`
7.	
8.	`public void userLeftOrRight(){`
9.	`Scanner keyboard = new Scanner(System.in);`
10.	`int i = 0;`
11.	`while(i < 1){`
12.	`System.out.println ("Would you like to take the left or right path?");`
13.	`System.out.println ("Press 1 for Left.");`
14.	`System.out.println ("Press 2 for Right.");`
15.	`if(keyboard.hasNextInt()){`
16.	`int userChoice = keyboard.nextInt();`

```
17.             }
18.                 if(userChoice == 1 || userChoice == 2){
19.                     if(userChoice == 1){
20.                             this.turnLeft();
21.                             this.move();
22.                             this.turnRight();
23.                     }
24.                     else{
25.                             this.turnRight();
26.                             this.move();
27.                             this.turnLeft();
28.                     }
29.                     i++;
30.                 }
31.                 else{
32.                     System.out.println ("Please enter a valid integer.");
33.                 }
34.             }
35.             else{
36.                 System.out.println ("Please enter a valid integer.");
37.             }
38.             keyboard.nextLine();
39.         }
40.     }
41. }
```

```
42.  public class Demo_1_ORIGINAL extends Object
43.  {
44.     public static void main(String[] args)
45.     {
46.        City seattle = new City();
47.        RobotIO bob = new RobotIO(seattle, 3, 2,
     Direction.EAST, 0);
48.
49.        new Wall(seattle, 3, 3, Direction.WEST);
50.        bob.userLeftOrRight();
51.        System.out.println ("You've made your choice.");
52.     }
53.  }
```

Example Trace:

In a nutshell, this technique allows you to accept user input to choose the robots direction, then process the information in one of four ways. The first trace is a trace of the user choosing option 1, take a left turn. The second is option 2, take a right turn. The third trace is a trace of the user choosing an integer that is not a valid option, and the fourth would be the programs operation if the user put in a data type other then an integer.

In order to go over these details more thoroughly, here is a (partial) trace of the above program, with some additional explanation afterward

Line#	Program Statement	Robot Info	i	userChoice	True / False	USER INPUT	Wall 1	Output
46	`public static void main(String[] args`	-	-					
48	`city seattle = new City(;`	-	-					
49	`RobotIO bob = new RobotIO(seattle, 3, 2, Direction.EAST, 0;`	3,2,E	-					
51	`new Wall(seattle, 3, 3, Direction.WEST`	3,2,E	-				3,3,W	

	;							
53	bob.userLeftOr Right(;	3,2,E	-				3,3,W	
12	public void userLeftOrRigh t({	3,2,E	-				3,3,W	
13	int i = 0;	3,2,E	0				3,3,W	
14	while(i < 1{	3,2,E	0		TRUE		3,3,W	
15	System.out.pri ntln ("Would you like to take the left or right path?";	3,2,E	0				3,3,W	Would you like to take the left or right path?
16	System.out.pri ntln ("Press 1 for Left.";	3,2,E	0				3,3,W	Press 1 for Left.
17	System.out.pri ntln ("Press 2 for Right.";	3,2,E	0				3,3,W	Press 2 for Right.
18	if(keyboard.ha sNextInt({	3,2,E	0		TRUE	1	3,3,W	
19	int userChoice = keyboard.nextI nt(;	3,2,E	0	1		1	3,3,W	
20	if(userChoice == 1 || userChoice == 2{	3,2,E	0	1	TRUE	1	3,3,W	
21	if(userChoice == 1{	3,2,E	0	1	TRUE	1	3,3,W	
22	this.turnLeft(;	3,2,N	0	1		1	3,3,W	
23	this.move(;	3,1,N	0	1		1	3,3,W	
24	this.turnRight (;	3,1,E	0	1		1	3,3,W	
31	i++;	3,1,E	1	1		1	3,3,W	
40	keyboard.nextL ine(;	3,1,E	1	-		1	3,3,W	
14	while(i < 1{	3,1,E	1	-	FALSE	-	3,3,W	
54	System.out.pri ntln ("You've made your choice.";	3,1,E	-	-		-	3,3,W	You've made your choice.

Option 2: User Input 2 (skipping 46-15)

Line #	Program Statement	Robot Info	i	user Choi	True / False	USER INPUT	Wall 1	Output

				ce				
15	System.out.println ("Would you like to take the left or right path?";	3,2,E	0				3,3,W	Would you like to take the left or right path?
16	System.out.println ("Press 1 for Left.";	3,2,E	0				3,3,W	Press 1 for Left.
17	System.out.println ("Press 2 for Right.";	3,2,E	0				3,3,W	Press 2 for Right.
18	if(keyboard.hasNextInt({	3,2,E	0		TRUE	2	3,3,W	
19	int userChoice = keyboard.nextInt(;	3,2,E	0	2		2	3,3,W	
20	if(userChoice == 1 || userChoice == 2{	3,2,E	0	2	TRUE	2	3,3,W	
21	if(userChoice == 1){	3,2,E	0	2	FALSE	2	3,3,W	
26	else{	3,2,E	0	2		**2**		
27	this.turnRight();	3,2,S	0	2		2		
28	this.move();	4,2,S	0	2		2		
29	this.turnLeft();	4,2,E	0	2		**2**		
31	i++;	4,2,E	1	2		2	3,3,W	
40	keyboard.nextLine(;	4,2,E	1	2		-	3,3,W	
14	while(i < 1{	4,2,E	1	2	FALSE	-	3,3,W	
54	System.out.println ("You've made your choice.";	4,2,E	-	-		-	3,3,W	You've made your choice.

Option 3: User Input non-valid integer (skipping 46-15)

Line #	Program Statement	Robot Info	i	userChoice	True / False	USER INPUT	Wall 1	Output
15	System.out.println ("Would you like to take the left or right path?";	3,2,E	0				3,3,W	Would you like to take the left or right path?
16	System.out.println ("Press 1 for Left.";	3,2,E	0				3,3,W	Press 1 for Left.
17	System.out.println ("Press 2 for Right.";	3,2,E	0				3,3,W	Press 2 for Right.

18	if(keyboard.hasNextInt({	3,2,E	0		TRUE	3	3,3,W	
19	int userChoice = keyboard.nextInt();	3,2,E	0	3		3	3,3,W	
20	if(userChoice == 1 || userChoice == 2{	3,2,E	0	3	FALSE	3	3,3,W	
33	else{	3,2,E	0	3		3	3,3,W	
33	System.out.println("Please enter a valid integer.");	3,2,E	0	3		3	3,3,W	Please enter a valid integer.
40	keyboard.nextLine();	3,2,E	0	3		-	3,3,W	
14	while(i < 1){	3,2,E	0	3	TRUE	-	3,3,W	
	->repeats until correct int is input							

Option 4: User Input non-integer (skipping 46-15)

Line #	Program Statement	Robot Info	i	userChoice	True / False	USER INPUT	Wall 1	Output
15	System.out.println ("Would you like to take the left or right path?";	3,2,E	0				3,3,W	Would you like to take the left or right path?
16	System.out.println ("Press 1 for Left.";	3,2,E	0				3,3,W	Press 1 for Left.
17	System.out.println ("Press 2 for Right.";	3,2,E	0				3,3,W	Press 2 for Right.
18	if(keyboard.hasNextInt({	3,2,E	0		FALSE	B	3,3,W	
37	else{	3,2,E	0			B	3,3,W	
20	if(userChoice == 1 || userChoice == 2{	3,2,E	0		FALSE	B	3,3,W	
33	else{	3,2,E	0			B	3,3,W	
37	System.out.println ("Please enter a valid integer.");	3,2,E	0			B	3,3,W	Please enter a valid integer.
40	keyboard.nextLine();	3,2,E	0			-	3,3,W	
14	while(i < 1){	3,2,E	0		TRUE	-	3,3,W	
	->repeats until							

	correct int is input							

You'll notice that the trace starts at beginning of the **main** function, on line 46, like normal. It proceeds normally until line 54, although it's worth noting that lines 4 and 48 are creating RobotSE robot which has turnRight already available, instead of the normal "plain vanilla" Robot robots.

Once our city is created, and our robot and wall placed, we call the command userLeftOrRight(); skipping back up to line 12, we can see what this command will do. First off, we create a new int, called i, this int is simply to keep track whether or not a valid option has been chosen. Next, we start a while loop, this loop will run until i's value is equal to one. We will set i's value to one after we accomplish having a valid option chosen by the user for the robot to follow.

We now print the options for the user, on 3 lines:

```
Would you like to take the left or right path?
Press 1 for Left.
Press 2 for Right.
```

Line 18 is checking whether the user has input an integer. There are 4 total possibilities here. The user can enter 1, 2, a valid but unused integer, or a different character type. The four routes the program will take are as follows:

1. The user does not input an integer, we skip down to line 37, the else to this if, and ask the user to `Please enter a valid integer.` then proceed to line 40, clear the users input with keyboard.nextLine(); and restart the while loop on line 14.

2. The user inputs any integer on line 18 we are taken to the next line, which creates the local variable userChoice, then sets it equal to the users input. Line 20 checks to see if the users input was equal to 1 or 2, if it is not, we are taken to the else line on 33, which again asks for a valid integer, then proceeds to clear the keyboard input and restart the while loop. Notice that we still have a variable with userChoice set, this should not matter as it will be recast during the next while loop.

3. The user inputs a 1, this passes the if statement. In the case a 1 is entered, line 21's if statement will be true, we we turn left, move up a spot, then turn right to the starting direction. Once this is complete, we skip down to line 31, increase the i counter by one, skip down to 40, clear the keyboard input, start the while loop which fails, and skip back down to main.

4.The user inputs a 2, this passes the if statement on 20, but fails the next if statement on line 21, so we proceed to the else on line 26. At this point we turn right, move one spot, turn back to the left to the starting direction. Once this is complete, we skip down to line 31, increase the i counter by one, skip down to 40, clear the keyboard input, start the while loop which fails, and skip back down to main.

When back in main, on line 55, and we print to the user `You've made your choice.` and end the program.

Syntax Explanation:

Integers are not the only data type that you can accept from the keyboard, but we will only be going over integers at this point.

Let's start with the program as it's written here. You will need to create both outcomes for accepting the proper integers that you have programmed the program to accept, as well as for options that are not accepted. You do not want the program to break if the user does not read the instructions correctly. Notice the 4 options we went over earlier here in the while loop.

Below, you can see the finished program with the new information highlighted in yellow.

```
import becker.robots.*;
import java.util.*;

class RobotIO extends RobotSE{
    public RobotIO(City c, int st, int ave, Direction dir,
int num){
        super(c, st, ave, dir, num);
    }

    Scanner keyboard = new Scanner(System.in);

    public void userLeftOrRight(){
        int i = 0;
        while(i < 1){
            System.out.println ("Would you like to take the
left or right path?");
            System.out.println ("Press 1 for Left.");
            System.out.println ("Press 2 for Right.");
            if(keyboard.hasNextInt()){
                int userChoice = keyboard.nextInt();
                if(userChoice == 1 || userChoice == 2){
                    if(userChoice == 1){
                            this.turnLeft();
                            this.move();
                            this.turnRight();
                    }
                    else{
                            this.turnRight();
                            this.move();
                            this.turnLeft();
                    }
                    i++;

                }
```

```
                else{
                    System.out.println ("Please enter a valid integer.");
                }
            }
            else{
                System.out.println ("Please enter a valid integer.");
            }
            keyboard.nextLine();
        }
    }
}
public class Demo_1_ORIGINAL extends Object
{
   public static void main(String[] args)
   {
      City seattle = new City();
      RobotIO bob = new RobotIO(seattle, 3, 2, Direction.EAST, 0);

      new Wall(seattle, 3, 3, Direction.WEST);

      bob.userLeftOrRight();
      System.out.println ("You've made your choice.");
   }
}
```

Help With The Logic:

This is good to use whenever you want to take user input to make a decision. The decisions you can accept are very broad, so we have chosen to just make the decision to turn right or left in this example. Since this choice is in its own re-callable command you can use the same method at any turn you would need during the program.

Important Details:

Point #1

If you do not set up the program to handle incorrect input, or to clear the keyboard after each while loop you can get stuck in a loop of running text, or crashing the program. Make sure to set up the program to handle whatever the user could screw up.

Point #2

It is important to use the command `keyboard.nextLine();` if you fail to do this, the program can also hang, or get stuck in an infinite loop.

Mistakes People Commonly Make (And How To Fix Them):

Getting stuck in a loop:

Detailed Example Of Error: If you do not set up the program clear the keyboard input at the end of the loop, you will get stuck in an infinite loops of the program asking for a valid integer, then giving you the input options, then reseting back to thinking the user has input the invalid response again.

Detailed Example Of Fix: (unless it's obvious): Make sure that no matter which option the program takes, (invalid integer, valid integer, non-integer), that the program clears the keyboard with keyboard.nextLine(); after taking the correct course.

Plagiarism

If you believe that some or all of this document infringes on your intellectual property (i.e., part or all of this document is copied from something you've written) please immediately contact Mike Panitz at Cascadia Community College (perhaps using the Faculty And Staff Directory at http://www.cascadia.edu/pages/searchtemplate.aspx)