ORACLE

Academy

Java Fundamentals

5-1:

Scanner and Conditional Statements

ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Use Scanner for user input during program execution
  - Use if-else logic and statements
  - Apply switch logic and statements in Java code
  - Use break and default effectively in a switch statement
  - Use the ternary operator

Prompting the User for Input: Scanner

- Keyboard input using a Scanner requires the following import statement:

```
import java.util.Scanner;
```

- Prompting the user can be done with simple code that will appear in the console screen where the user can then enter their input

```
System.out.println("Write instructions for user here.");
```


When you are new to programming you will sometimes forget this and then wonder why the program is "hung" when it is just waiting for user input.

Scanner

- To read in the input that the user has entered, use the Java object Scanner
- To initialize a Scanner, write:

```
Scanner in = new Scanner(System.in);
```

Scanner is commonly named "in" which is short for "input"

System.in is used for the scanner to read the users' input from the console screen.

Technically, System.in is a file that is associated with the keyboard or "standard input". Scanner can be used with File objects to read from a text file.

Why Scanner?

- Scanner makes it easy to read in the user's input because it already has methods that do this very task
- The Scanner method next() reads in the user's input as a String and returns that String
- This line of code:
  - Creates a new string called input
  - Scans in the string that the user has entered into the output console using the scanner called in
  - Sets input equal to the string that was read in by the scanner

```
String input = in.next();
```

Scanner's nextInt() Method

- The Scanner method nextInt() reads in the user's input as an integer and returns that integer
- This line of code creates a new int, called answer

```
import java.util.Scanner;
public class InputExample{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      System.out.println("Enter your name:");
      String name = in.next(); //reads in text until a space
      System.out.println("Enter a number:");
      int answer = in.nextInt(); //reads an integer value
      System.out.println(name + ", the number you entered is: "
                         + answer);
   }//end method main
}//end class InputExample
```


As a programmer you will need to write code that will handle situations where invalid data types are entered, i.e., "twenty" when an int value of 20 is expected. Professional programs will have methods that test data for obvious and not so obvious invalid data. This is sometimes referred to as "bullet proofing" the code.

More Useful Scanner Methods

Method	What It Does	When to Use
nextInt()	Similar to next(), this function reads in the user's input and returns it's integer value	When you prompt the user for an integer value and wish to read in the user's input as an integer rather than as a string
hasNext()	Returns true if the scanner has another input, and false otherwise	When you wish to know if there is any more input for the scanner to read in
close()	Closes the scanner	When you are done reading in input, it is good practice to close the scanner, especially when reading input from the console screen This keeps the program from running continuously The scanner may expect more input if it is never closed

hasNext() and close() are used for reading files

Not closing Scanner objects or other files can lead to tied up resources or even to corrupted files. Always close Scanner objects as soon as possible.

Relational Operators

- Java has six relational operators used to test primitive or literal numerical values
- Relational operators are used to evaluate if-else and loop conditions

Relational Operator	Definition
>	Greater than
>=	Greater than or equal to
==	Equal to
<	Less than
<=	Less than or equal to
!=	Not equal to

In Java, a common syntax error is to use = when == was intended. You should read = not as "equals" but rather, "is assigned".
For more information on these operators as well as those on slide 12, review the precedence table at https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Relational Operators Example

- Values are tested on either side of the operator and a true or false value is returned
- This value can be stored or used as part of a control structure to control program flow
- In this example, the variable madeHonorRoll is assigned a true value when the expression grade >= 88 evaluates as true

```
int grade = 99;
boolean madeHonorRoll = grade >= 88;
if(madeHonorRoll)
   System.out.println("You made the Honor Roll.");
```

Relational Operators Example

- The same example can be evaluated with the use of the boolean variable
- However, the expression grade >=88 evaluates as true or false depending on the value assigned to grade
- Boolean values are necessary as a condition in an if-else statement or loop

```
int grade = 99;

if(grade >= 88)
      System.out.println("You made the Honor Roll.");
```

Logic Operators

- Java has three logic operators used to combine boolean expressions into complex tests

Logic Operator	Meaning
&&	AND
||	OR
!	NOT

Logic Operators Example 1

- In this example, the phrase "You qualify for the scholarship" will print if both conditions are true
- For the message to print, madeHonorRoll must be true and the numberDaysAbsent must be equal to zero

```
int numberDaysAbsent = 0;
int grade = 99;
boolean madeHonorRoll = grade >= 88;

if(madeHonorRoll && numberDaysAbsent==0)
   System.out.println("You qualify for the scholarship.");
```

Logic Operators Example 2

- Describe the results for each of the following Java code segments

```
double grade=65;
int numDaysAbsent=2;
boolean madeHonorRoll = grade >= 88;
if(!madeHonorRoll && numDaysAbsent<3)
   System.out.println("You qualify for free tutoring help.");
if(grade > 70 && numDaysAbsent < 5)
  System.out.println("You may try out for the sports team.");
```

Logic Operators Example 2 Solution

- Results for each of the following Java code segments:
  - The phrase "You qualify for free tutoring help." prints to the screen
  - However the phrase "You may try out for the sports team." does not print as the student's grade is not above 70

```
double grade=65;
int numDaysAbsent=2;
boolean madeHonorRoll = grade >= 88;
if(!madeHonorRoll && numDaysAbsent<3)
   System.out.println("You qualify for free tutoring help.");
if(grade > 70 && numDaysAbsent < 5)
  System.out.println("You may try out for the sports team.");
```

Syntax for if-else Statements

- To build an if-else statement, remember the following rules:
  - An if-else statement needs a condition or method that is tested for true/false
  - For example:

```
if(y > 17)

if(x == 5)

if(s1.equals(s2))
```

You have already came across the if-else statement in both the Alice and Greenfoot sections.

Syntax for if-else Statements

- Likewise, an optional else if statement can be tested, for example:

```
if(y > 17){
 System.out.println("y > 17");
else if(y == 17)
 System.out.println("y == 17");
```

- The optional else statement will take care of every other possibility

```
if(y >= 17){
 System.out.println("y >= 17");
else if(y == 7)
 System.out.println("y >= 17");
else
 System.out.println("y < 17");
```

if-else Statements with the char Data Type

```
import java.util.Scanner;
public class Calculator{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int answer = 0;
      System.out.println("Enter a number: ");
      int num1 = in.nextInt();
      System.out.println("Enter another number: ");
      int num2 = in.nextInt();
      System.out.println("Enter the operand(* / % + -): ");
      char input = in.next().charAt(0);
      if( input == '*' )
         answer = num1 * num2;
      else if( input == '/' )
         answer = num1 / num2;
      else if( input == '%' )
         answer = num1%num2;
      else if( input == '+' )
         answer = num1 + num2;
      else if( input == '-' )
         answer = num1 - num2;
      else
         System.out.println("Invalid Command");
      System.out.println("The result is: " + answer);
    }//end method main
}//end class Calculator
```


If there is more than one statement for the if or else, they must be enclosed in {braces}. The following is a common logic error:

```
if( false )
  System.out.println("I never print");
  System.out.println("I always print");
```

if-else Statements with the int Data Type

- It can be useful to use print instead of println when prompting the user, keeps the cursor on the same line:

```
import java.util.Scanner;
public class ValueChecker{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int value = 0;
      System.out.print("Enter a number:");
      value = in.nextInt();
      if( value == 7)
         System.out.println("That's lucky!");
      else if( value == 13)
         System.out.println("That's unlucky!");
      else
         System.out.println("That is neither lucky nor unlucky!");
    }//end method main
}//end class ValueChecker
```

if-else Statements with the String Data Type

- nextLine() reads in all text regardless of spaces

```
import java.util.Scanner;
public class StringChecker{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      String name = "";
      System.out.print("Enter your name:");
      name = in.nextLine();
      if(name.equals("Elvis"))
         System.out.println("You are the king of Rock and"
                            + " Roll");
      else if(name.equals("Michael Jackson"))
         System.out.println("You are the king of pop!")
      else
         System.out.println("You are not the king!");
    }//end method main
}//end class StringChecker
```

Switch Statement

- Like the if-else example earlier, consider a program that takes two integer inputs from a user and performs a specified mathematical operation
- To support different operators a test is needed to see if the input was any of the following:
  - `* / + - %`
- How would you check to see what the user typed?

Switch Statement Changes Program Flow

- A switch statement is another way of changing program flow depending on the input value

```
import java.util.Scanner;
public class Calculator{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int answer = 0;
      System.out.print("Enter a number: ");
      int num1 = in.nextInt();
      System.out.print("Enter another number: ");
      int num2 = in.nextInt();
      System.out.println("Enter the operand: ");
      char input = in.next().charAt(0);
                                                    ...
```

Switch Statement Changes Program Flow

```
...
      switch (input){
         case '*' : answer = num1 * num2;
                    break;
         case '/' : answer = num1 / num2;
                    break;
         case '%' : answer = num1 % num2;
                    break;
         case '+' : answer = num1 + num2;
                    break;
         case '-' : answer = num1 - num2;
                    break;
         default: System.out.println("Invalid Command.");
      }//end switch
      System.out.println("The result is: " + answer);
   }//end method main
}//end class Calculator
```


The default:, like the optional else in an if-else statement, is also optional.
Remember that you use a : for the case statements not a ; as that represents the end of a statement.
The switch is preferred over nested if-else because it is easier to read. However, the switch does not work well with ranges:

```
if(age >=13 && <=19)
  System.out.println("Teenager");
```

as opposed to:

```
switch(age)
  case 13:
  case 14:
  case 15:
  case 16:
  case 17:
  case 18:
  case 19:
     System.out.println("Teenager");
     break;
```

Switch Statement Keywords

- A switch statement uses 3 keywords: switch, case, and default
  - switch:
    - specifies which variable to test for value
  - case:
    - compares the value of the switch variable
  - default:
    - when the input does not match any of the cases, the compiler chooses the default action (like else in a list of if statements)

Additional Information about Switch Statements

- After each case, include the keyword break
- If not included, the code will "fall through" and execute each case until break is encountered
- In Java SE 7 and later, you can use a String object in the switch statement's expression

Switch Statement Example 1

- This example shows how "fall through" behavior works For membership sales, the more memberships sold, the more prizes a sales rep wins for those sales

Switch Statement Example 1

```
import java.util.Scanner;
public class SalesWinners {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);

      System.out.println("How many memberships did you sell?");
      int sales = in.nextInt();

      switch(sales){
         case 6:  System.out.println("You win $1000");
         case 5:  System.out.println("You win a Samsung Galaxy");
         case 4:  System.out.println("You win Laptop");
         case 3:  System.out.println("You win iPod");
         case 2:  System.out.println("You win Stapler");
         case 1:  System.out.println("You win Staple Remover");
                  break;
         default:  System.out.println("No Gift");
      }//end switch
   }//end method main
}//end class SalesWinners
```

Switch Statement Example 2

- A 9th year student in high school is considered a freshman, 10th year students are sophomores, etc

```
import java.util.Scanner;
public class ClassYear {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.println("What grade are you in?");
      int grade = in.nextInt();
      switch(grade){
         case 9: System.out.println("You are a freshman");
                 break;
         case 10:  System.out.println("You are a sophomore");
                 break;
         case 11:  System.out.println("You are a junior");
                                  break;
         case 12: System.out.println("You are a senior");
                 break;
          default: System.out.println("Invalid grade");
      }//end switch
   }//end method main
}//end class ClassYear
```

Switch Statement Example 3

- Given a month and year, the number of days in the month are calculated
- Encourage students to research "Leap Year" rules

```
import java.util.Scanner;
public class LeapYearCalculator {
   public static void main(String[] args) {
       Scanner in = new Scanner(System.in);
       System.out.println("Enter the month");
       int month = in.nextInt();
       System.out.println("Enter the year");
       int year = in.nextInt();
                                                    ...
```

Switch Statement Example 3

```
...
      switch(month){
         case 4:
         case 6:
         case 9:
         case 11: System.out.println("That month has 30 days");
               break;
         case 1:
         case 3:
         case 5:
         case 7:
         case 8:
         case 10:
         case 12:  System.out.println("That month has 31 days");
                   break;
                                                              ...
```

Switch Statement Example 3

```
...
         case 2:    if((year % 4 == 0 && year % 100 != 0) ||
                        year % 400 == 0)
                      System.out.println("That month has 29 days");
                    else
                      System.out.println("That month has 28 days");
                    break;
         default:     System.out.println("Invalid Input");
       }//end switch
   }//end method main
}//end class LeapYearCalculator
```


case 2: takes into account leap year rules implemented with the Gregorian calendar. A leap year every 4 years gives about 3 too many leap days every 400 years. Thus, while 2000 was a leap year, 1800 and 1900 were not and 2100 will not be.
For fun, go to http://www.timeanddate.com and enter the year 1752 and the month of September. This was the year that the United Kingdom converted from the Julian calendar to the Gregorian calendar. The missing days were to get the seasons back into alignment with the calendar. Other countries adopted the Gregorian calendar in different years with Greece being one of the last to change in 1923.

Switch Expressions

- Since Java 14 a switch expression has been included in the Java language
- The switch label is now written as: case value -> code;
- Only the code to the right of the label is going to be executed if the label is matched
- After each case, the keyword break is not required
- The cases of a switch expression must be exhaustive, all possible values must have a matching switch label
- A default is normally required

Switch Expressions Example 1

- You can rewrite the switch 2 example using the new switch expression labels
- Compare both ways to review the differences

```
import java.util.Scanner;
public class ClassYear {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.println("What grade are you in?");
      int grade = in.nextInt();
      switch(grade){
         case 9  -> System.out.println("You are a freshman");
         case 10 -> System.out.println("You are a sophomore");
         case 11 -> System.out.println("You are a junior");
         case 12 -> System.out.println("You are a senior");
         default -> System.out.println("Invalid grade");
      }//end switch
   }//end method main
}//end class ClassYear
```

Ternary Operator

- The ternary operator (?:) in Java is used to create a shorter version of an if-else statement
- In the following example, there are three parameters using this operator
  - The first is the boolean test (c>9)
  - The second (6) is the value to return if the test is true
  - The third (7) is the value to return if the test is false
  - It is often used as part of an assignment

```
int x = c > 9 ? 6 : 7;
//If c is greater than 9, x is 6; else x is 7
```


The ternary operator is sometimes called the conditional operator. It is the only operator in Java to have 3 operands.

Ternary Operator Example

- Here, an if-else statement is used to check for String equality

```
String s1 = "Hello";
String s2 = "Goodbye";
if(s1.equals(s2))
   System.out.println("Yes");
else
   System.out.println("No");
```

- A similar result is achieved using the ternary operator

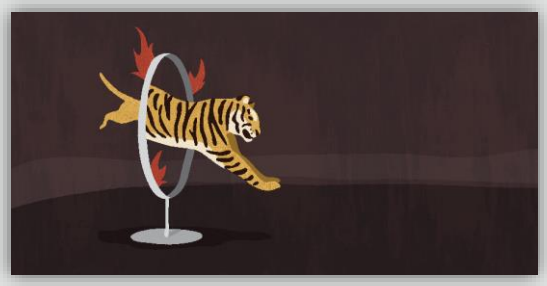
```
String s1 = "Hello";
String s2 = "Goodbye";
String answer = s1.equals(s2) ? "Yes" : "No";
System.out.println(answer);
```

Terminology

- Key terms used in this lesson included:
  - if statements
  - If-else statements
  - Scanner
  - switch statements (case, switch, and default)
  - Ternary operators

Summary

- In this lesson, you should have learned how to:
  - Use Scanner for user input during program execution
  - Use if-else logic and statements
  - Apply switch logic and statements in Java code
  - Use break and default effectively in a switch statement
  - Use the ternary operator




ORACLE
Academy