SENG1110/SENG6110 Object Oriented Programming

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

www.newcastle.edu.au

Lecture 3

Flow of control – selection

Outline

- Previously…
  - Variables, expressions and operators
  - Input and output
  - The Class String
  - Documentation and Style
  - Examples
- Now…
  - Conditional statements
    - `if` and `if-else`
    - Compare – primitive x class types
    - Type boolean
    - `Switch` statement
  - Java API documentation
  - Program errors
  - Input/Output – TIO and GUI

…attention…☺

- Week 3 is THE WEEK!!!!!
- It is **vital** now you start
  - to do a lot of different exercises:
    - From the book
    - Computer lab exercises
- Ask help:
  - PASS
  - Help Desk

Example - SMS cost pseudocode

```
input number_messages;
total = 10 + 0.22*number_messages
output total
```

Example – SMS – using TIO input/output

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

        int count;
        double cost;

        System.out.print(“Input No of Messages: “);
        count = console.nextInt();
        cost = 10 + 0.22 * count;
        System.out.print(“Total Cost is “+cost);
   }
}
```

Example - SMS Cost modification 1

- Suppose that in the SmsCost example you want to include the follow aspect: if the total cost is more than A$50 then the user will receive a discount of 5%. How to do this?
- Using conditional statement - `if` and `if-else`

`if` and `if-else` - syntax

```
if (condition)
{
        statement;
        statement;
        …;
}
```

`if` and `if-else` - syntax

```
if (condition)
{
        statement;
        statement;
        …;
}
else
{
        statement;
        statement;
        …;
}
```

`if` and `if-else` - syntax

- The braces may be omitted if a single statement follows the if or else

```
if (condition)
     statement;
else
     statement;
```

```
if (condition)
   statement;
else
{
     statement;
     statement;
}
```

avoid

```
if (condition)
{
    statement;
    statement;
}
else
    statement;
```

avoid

SMS cost example 1

- Include the follow aspect: if the total cost is more than A$50 then the user will receive a discount of 5%. How to do this?

```
input number_messages;
total = 10 + 0.22*number_messages
if (total > 50)
{
   total = total*0.95
}
output total
```

SMS cost example 1

```
import java.util.*;
public class SMScost
{
  public static void main (String[] args)
  {
    Scanner console = new Scanner(System.in);
    int count;
    double cost;

    System.out.print("Input No of Messages: ");
    count = console.nextInt();
    cost = 10 + 0.22 * count;
    if (cost>50)
         cost = cost*0.95;

    System.out.print("Total Cost is "+cost);
  }
}
```

SMS cost example 1

```
import java.util.*;
public class SMScost
{
  public static void main (String[] args)
  {
    Scanner console = new Scanner(System.in);
    int count;
    double cost;

    System.out.print("Input No of Messages: ");
    count = console.nextInt();
    cost = 10 + 0.22 * count;
    if (cost>50)
    {
         cost = cost*0.95;
         System.out.print(" You received a 5% discount ");
    }
    System.out.print("Total Cost is "+cost);
  }
}
```

SMS cost example 2

- Suppose that in the SmsCost example you want to include the follow aspect:
  - if the total cost is more than A$50 then the user will receive a 5% discount.
  - If total cost is less or equal than A$50 then the discount will be 2%.

SMS cost example 2

```
input number_messages;
totalcost = 10 + 0.22*number_messages
if (totalcost > 50)
   totalcost = totalcost*0.95
else
   totalcost = totalcost*0.98
output totalcost
```

SMS cost example 2

```
import java.util.*;
public class SMScost
{
  public static void main (String[] args) throws IOException
  {
   Scanner console = new Scanner(System.in);
   int count;
   double cost;

   System.out.print("Input No of Messages: ");
   count = console.nextInt();
   cost = 10 + 0.22 * count;
   if (cost>50)
        cost = cost*0.95;
    else
        cost = cost*0.98;

   System.out.print("Total Cost is "+cost);
  }
}
```

SMS cost example 3

- New customer request:
  - Change the SMS program so that it can also compute a more complicated cost formula, eg. If the number of messages is more than 50, then subsequent messages are given a 10% discount.

SMS cost example 3

- Pseudocode would be:

```
if (count > 50)
   cost = 10 + 0.22*50 + 0.22*(count-50)*0.9
else
   cost = 10 + 0.22 * count

output cost

We could alter the first formula to:
         10 + 0.22*(50 + (count-50)*0.9)
or       21.0 + (count-50)*0.198
or even  11.1 +  count*0.198
```

SMS cost example 3

```
import java.util.*;
public class SMScost
{
  public static void main (String[] args)
  {
   Scanner console = new Scanner(System.in);
   int count;
   double cost;

   System.out.print("Input No of Messages: ");
   count = console.nextInt();
   if (count > 50)
        cost = 10 + 0.22*50 + 0.22*(count-50)*0.9;
   else
        cost = 10 + 0.22*count;

   System.out.print("Total Cost is "+cost);
  }
}
```

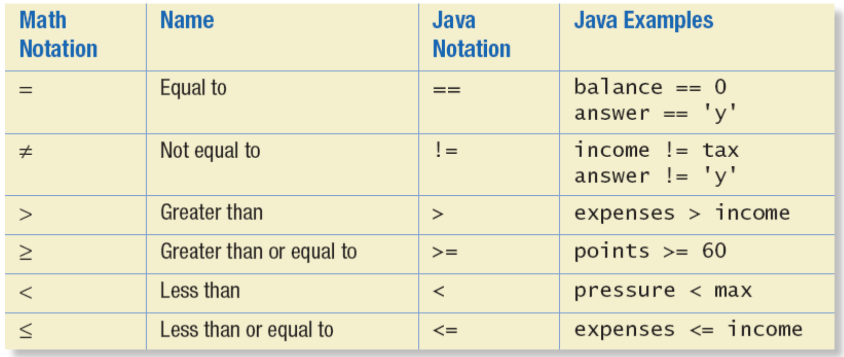
Introduction to Boolean Expressions

- The value of a boolean expression is either

true
or
false

- Examples
  - `time < limit`
  - `balance <= 0`

Java comparison operators



Math Notation	Name	Java Notation	Java Examples

=	Equal to	`==`	`balance == 0`
			`answer == 'y'`

≠	Not equal to	`!=`	`income != tax`
			`answer != 'y'`

>	Greater than	`>`	`expenses > income`
≥	Greater than or equal to	`>=`	`points >= 60`
<	Less than	`<`	`pressure < max`
≤	Less than or equal to	`<=`	`expenses <= income`

Compound boolean expressions

Name	Java Notation	Java Examples
Logical *and*	&&	`(sum > min) && (sum < max)`
Logical *or*	`||`	`(answer == 'y') || (answer == 'Y')`
Logical *not*	!	`!(number < 0)`

Boolean Operators

- The Effect of the Boolean Operators `&&` (and), `||` (or), and `!` (not) on Boolean values

Value of *A*	Value of *B*	Value of *A* && *B*	Value of *A* || *B*	Value of ! (*A*)
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Boolean expressions - examples

Expression	Value	Explanation
!('A' > 'B')	**true**	Because 'A' > 'B' is **false**, !('A' > 'B') is **true**
!(6 <= 7)	**false**	Because 6 <= 7 is **true**, !(6 <= 7) is **false**.
(14 >= 5) && ('A' < 'B')	**true**	Because (14 >= 5) is **true**, ('A' < 'B') is **true**, and **true** && **true** is **true**, the expression evaluates to **true**.
(24 >= 35) && ('A' < 'B')	**false**	Because (24 >= 35) is **false**, ('A' < 'B') is **true**, and **false** && **true** is **false**, the expression evaluates to **false**.

Boolean expressions - examples

Expression	Value	Explanation
(14 >= 5) || ('A' > 'B')	**true**	Because (14 >= 5) is **true**, ('A' > 'B') is **false**, and **true** || **false** is **true**, the expression evaluates to **true**.
(24 >= 35) || ('A' > 'B')	**false**	Because (24 >= 35) is **false**, ('A' > 'B') is **false**, and **false** || **false** is **false**, the expression evaluates to **false**.
('A' <= 'a') || (7 != 7)	**true**	Because ('A' <= 'a') is **true**, (7 != 7) is **false**, and **true** || **false** is **true**, the expression evaluates to **true**.

Boolean expressions - examples

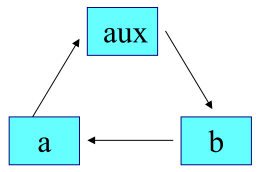
- Examples with operators

	a=1 b=2	a=1 b=4	a=1 b=3	a=6 b=2	a=6 b=4	a=6 b=3	a=5 b=2	a=5 b=3
`if ((a>5) && (b<3))`	F	F	F	T	F	F	F	F
`if ((a>5) || (b<3))`	T	F	F	T	T	T	T	F
`if ((a==5) && (b<3))`	F	F	F	F	F	F	T	F
`if ((a!=5) || (b<3))`	T	T	T	T	T	T	T	F
`if ((a==5) && (b!=3))`	F	F	F	F	F	F	T	F
`if ((a!=5) || (b!=3))`	T	T	T	T	T	T	T	F
`if ((a!=5) && (b!=3))`	T	T	F	T	T	F	F	F

Example – swap 2 variables

- Read 2 values, a and b, swap 2 variables if a is greater than b

```
input a
input b
if (a>b)
{
        aux=a;
        a=b;
        b=aux;
}
```



- What problems can occur if you don’t do the right sequence of these statements?

Example – swap 2 variables – Java code

```
import java.util.*;

public class SMSTio {

   public static void main (String[] args) {
        Scanner console = new Scanner(System.in);
        int a,b,aux;

        System.out.print("Input values to a and b “);
        a = console.nextInt();
        b = console.nextInt();
        if (a>b){
                aux=a;
                a=b;
                b=aux;
        }
        System.out.print(" a= “+a+”b=“+b);
   }
}
```

Nested if

```
if (condition)
{
    statement;
   if (condition)
    {
            statement;
            statement;
    }
   else
   {
            statement;
            statement;
    }
else
{
  statement;
  statement;
}
```

Nested if - Example

- Suppose the code below to decide whether credit is approved:

```
if ( name ok )
    if ( id ok )
       if ( amount ok )
          display credit approved
       else
          display credit denied
    else
       display credit denied
 else
    display credit denied
```

Nested if - Example

- Here is a more easily readable (and understandable) way to achieve the same thing (note the use of brackets around every Boolean expression):

```
if ( (name ok) && (id ok) && (amount ok) )
     display credit approved
else
     display credit denied
```

- How to rewrite using the boolean expression for

```
if (?????)
     display credit denied
else
     display credit approved
```

Nested if Example

```
if(balance > 50000.00)                  //Line 1
   interestRate = 0.07;                 //Line 2
else if(balance >= 25000.00)            //Line 3
        interestRate = 0.05;            //Line 4
else if(balance >= 1000.00)             //Line 5
        interestRate = 0.03;            //Line 6
else                                    //Line 7
    interestRate = 0.00;                //Line 8
```

Nested if Example

```
if(score >= 90)      System.out.println("The grade is A");
else if(score >= 80) System.out.println("The grade is B");
else if(score >= 70) System.out.println("The grade is C");
else if(score >= 60) System.out.println("The grade is D");
else                 System.out.println("The grade is F");
```

Nested Statements

- Different forms

First Form

```
if (a > b)
{
     if (c > d)
          e = f
}
     else
          g = h;
```

Second Form

```
if (a > b)
    if (c > d)
          e = f
     else
          g = h;
```

SMS cost example 4

- Is the previous Sms program ok?
- The program accepts negative numbers.
- How to correct this?

SMS cost example 4

```
input count
if (count>= 0)
{
    if (count > 50)
         cost = 10 + 0.22*50 + 0.22*(count-50)*0.9
    else
         cost = 10 + 0.22 * count
}
else
{
   output error message
}
output cost
```

SMS cost example 4

```
import java.util.*;
public class SMScost {
    public static void main (String[] args){
        Scanner console = new Scanner(System.in);
        int count;
        double cost;
        System.out.print("Input No of Messages: ");
        count = console.nextInt();

        if (count>=0)
        {
           if (count > 50)
                cost = 10 + 0.22*50 + 0.22*(count-50)*0.9;
           else
                cost = 10 + 0.22*count;
           System.out.print("Total Cost is "+cost);
        }
        else
        {
            System.out.print("Number of messages can't be negative");
        }
    }
}
```

Using `==`

- `==` is appropriate for determining if two integers or characters have the same value.
  `if (a == 3)`
  where `a` is an integer type
- `==` is **not** appropriate for determining if two floating points values are equal. Use `<` and some appropriate tolerance instead.
  `if (abs(b - c) < epsilon)`
  where `b`, `c`, and `epsilon` are floating point types

Using `==`

- `==` is not appropriate for determining if two objects have the same value.
  - `if (s1 == s2),` where `s1` and `s2` refer to strings, determines only if s1 and s2 refer the a common memory location.
  - If `s1` and `s2` refer to strings with identical sequences of characters, but stored in different memory locations, `(s1 == s2)` is false.

Using `==`

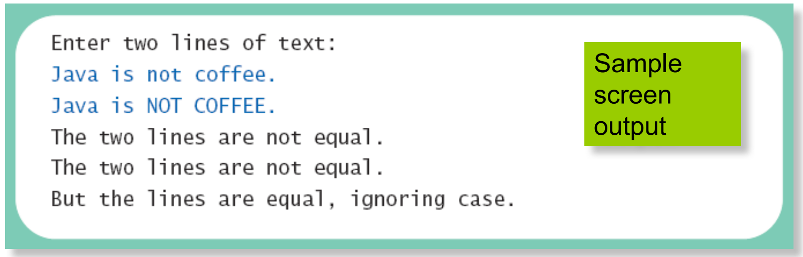
- To test the equality of objects of class String, use method `equals`.
  `s1.equals(s2)`
  or
  `s2.equals(s1)`
- To test for equality ignoring case, use method `equalsIgnoreCase`.
  `("Hello".equalsIgnoreCase("hello"))`

`equals` and `equalsIgnoreCase`

- Syntax

```
String.equals(Other_String)
String.equalsIgnoreCase(Other_String)
```

Testing Strings for Equality



```
Enter two lines of text:
Java is not coffee.
Java is NOT COFFEE.
The two lines are not equal.
The two lines are not equal.
But the lines are equal, ignoring case.
```

Sample screen output

```
import java.util.Scanner;
public class StringEqualityDemo
{
   public static void main (String [] args)
   {
        String s1, s2;
        System.out.println ("Enter two lines of text:");
        Scanner keyboard = new Scanner (System.in);
        s1 = keyboard.nextLine ();
        s2 = keyboard.nextLine ();
        if (s1.equals(s2))
            System.out.println ("The two lines are equal.");
        else
            System.out.println ("The two lines are not equal.");
        if (s2.equals(s1))
            System.out.println ("The two lines are equal.");
        else
            System.out.println ("The two lines are not equal.");
        if (s1.equalsIgnoreCase(s2))
            System.out.println ("But the lines are equal, ignoring case.");
        else
            System.out.println ( "Lines are not equal, even ignoring case.");
    }
 }
```

Method `compareTo`

- Syntax

  `String_1.compareTo(String_2)`

- Method `compareTo` returns
  - a negative number if `String_1` precedes `String_2`
  - zero if the two strings are equal
  - a positive number of `String_2` precedes `String_1`.

Java API Documentation

- Java API (application programming interface) Documentation

  http://docs.oracle.com/javase/8/docs/api/

The Conditional Operator

```
if (n1 > n2)
    max = n1;
else
    max = n2;
```

can be written as

```
max = (n1 > n2) ? n1 : n2;
```

- The `?` and `:` together are call the *conditional operator* or *ternary operator.*

The Conditional Operator

- The conditional operator is useful with print and println statements.

```
System.out.print("You worked " +
    ((hours > 1) ? "hours" ; "hour"));
```

The `exit` Method

- Sometimes a situation arises that makes continuing the program pointless.
- A program can be terminated normally by `System.exit(0).`

The `exit` Method

- Example

```
if (numberOfWinners == 0)
{
    System.out.println ("Error: Dividing by zero.");
    System.exit (0);
}
else
{
    oneShare = payoff / numberOfWinners;
    System.out.println ("Each winner will receive $"
  + oneShare);
}
```

The Type `boolean`

- The type **`boolean`** is a primitive type with only two values: **`true`** and **`false`**.
- Boolean variables can make programs more readable.

```
if (systemsAreOK)
```
instead of
```
if((temperature <= 100) && (thrust >= 12000)
  && (cabinPressure > 30) && …)
```

Boolean Expressions and Variables

- Variables, constants, and expressions of type **`boolean`** all evaluate to either **`true`** or **`false`**.
- A boolean variable can be given the value of a boolean expression by using an assignment operator.

```
boolean isPositive = (number > 0);
...
if (isPositive) ...
```

Naming Boolean Variables

- Choose names such as **`isPositive`** or **`systemsAreOk`**.
- Avoid names such as **`numberSign`** or **`systemStatus`**.

Short-circuit Evaluation

- Sometimes only part of a boolean expression needs to be evaluated to determine the value of the entire expression.
  - If the first operand associated with an **`||`** is **`true`**, the expression is **`true`**.
  - If the first operand associated with an **`&&`** is **`false`**, the expression is **`false`**.

Short-circuit Evaluation

- Short-circuit evaluation is not only efficient, sometimes it is essential!

- A run-time error can result, for example, from an attempt to divide by zero.

```
if ((number != 0) && (sum/number > 5))
```

switch Structures

```
switch(expression)
{
case value1: statements1
             break;
case value2: statements2
             break;
   ...
case valuen: statementsn
             break;
default: statements
}
```

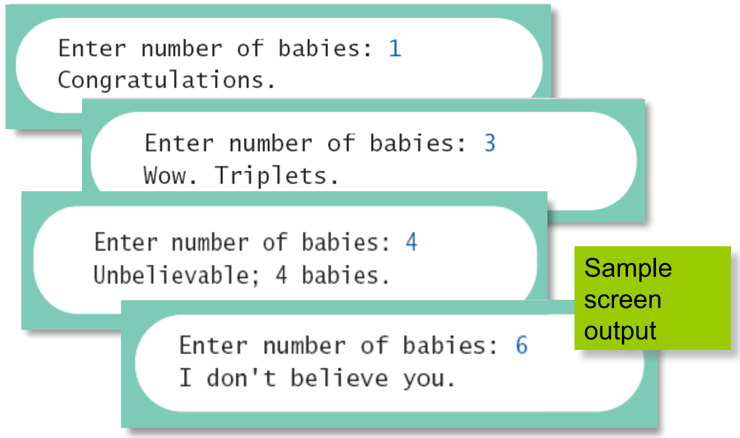
- Expression also known as selector
- Expression can be identifier
- Value can only be integral
- The action for each case ends with the word **break**

- Java 7 allows String expressions

switch Statement Example

```
switch(grade)
{
case 'A': System.out.println("The grade is A.");
          break;
case 'B': System.out.println("The grade is B.");
          break;
case 'C': System.out.println("The grade is C.");
          break;
case 'D': System.out.println("The grade is D.");
          break;
case 'F': System.out.println("The grade is F.");
          break;
default:  System.out.println("The grade is invalid.");
}
```

The switch Statement - example



```
Enter number of babies: 1
Congratulations.
```

```
Enter number of babies: 3
Wow. Triplets.
```

```
Enter number of babies: 4
Unbelievable; 4 babies.
```

```
Enter number of babies: 6
I don't believe you.
```

```
import java.util.Scanner;
public class MultipleBirths
{
   public static void main (String [] args)
   {
      int numberOfBabies;
      System.out.print ("Enter number of babies: ");
      Scanner keyboard = new Scanner (System.in);
      numberOfBabies = keyboard.nextInt ();
      switch (numberOfBabies)
      {
        case 1:  System.out.println ("Congratulations.");
                 break;
        case 2:  System.out.println ("Wow. Twins.");
                 break;
        case 3:  System.out.println ("Wow. Triplets.");
                 break;
        case 4:
        case 5:  System.out.print ("Unbelieveable; ");
                 System.out.println (numberOfBabies + " babies.");
                 break;
        default: System.out.println ("I don't believe you.");
                 break;
      }
   }
}
```

Program Errors

- **Syntax errors**
  - Occur when a syntax rule is violated
  - Are detected at compile time.
  - When the Java compiler finds a syntax error, it prints an error message.

- Example

```
System.ot.print(“Input No of Messages: “);
```

Program Errors

- **Run-time errors**
  - Occur when the computer is asked to do something that it considers illegal
  - Example

```
double z,x=1,y=0;
z = x/y;
```

  - A expression depends on the values contained in the variables.
  - The Java run-time environment will print a message telling us the nature of the error and where it was encountered.

Program Errors

- **Logic errors** (design errors or bugs)
  - Occur when we fail to express ourselves accurately.
    - •The instruction is phrased properly, and thus the syntax is correct.
    - •The instruction is meaningful, and thus the semantics are valid.
    - •But the instruction does not do what we intended, and thus is logically incorrect.
  - Programming environments do not detect logic errors automatically.

Debugging

- A bug is not easy to locate.
- You can try to find the bug(s):
  - Using pencil and paper
  - Using the code, adding extra lines to the program.
  - Using tools available (we will use BlueJ soon in the computer labs)
- Determining if any of the variables deviate from their expected values will highlight the existence of a bug.
- A variables value is printed in the terminal window as follows:

```
System.out.print ("<some message>" + <variable name>);
```

Debugging

```
import java.util.*;
public class Value
{
    public static void main (String[] args)
    {
        Scanner console = new Scanner(System.in);
        int x,y,z;

        System.out.print("Please Enter a value for x: ");
        x = console.nextInt();
        System.out.print("Please Enter a value for y: ");
        y = console.nextInt();
        System.out.print("Please Enter a value for z: ");
        z = console.nextInt();

        y = (x+y)/z + x;
        System.out.print("y = " + y);
    }
}
```

Debugging – Example 1

- Let's see the logical part

```
System.out.print("Please Enter a value for x: ");
x = console.nextInt();
System.out.print("Please Enter a value for y: ");
y = console.nextInt();
System.out.print("Please Enter a value for z: ");
z = console.nextInt();

y = (x+y)/z + x;
System.out.print("y = " + y);
```

- Suppose you have

x	y	z
2	3	2

x+y = 5
(x+y)/z = 2.5
(x+y)/z + x = 4.5

But the program gives 4. Why? How to discover what is happening?

Debugging – Example 1

- Modify the code

```
System.out.print("Please Enter a value for x: ");
x = console.nextInt();
System.out.print("Please Enter a value for y: ");
y = console.nextInt();
System.out.print("Please Enter a value for z: ");
z = console.nextInt();

int a = x+y;
System.out.print("a = "+a);   -> 5 -> ok

a = a/z;                     -> 2 -> not ok! What is the problem?
```

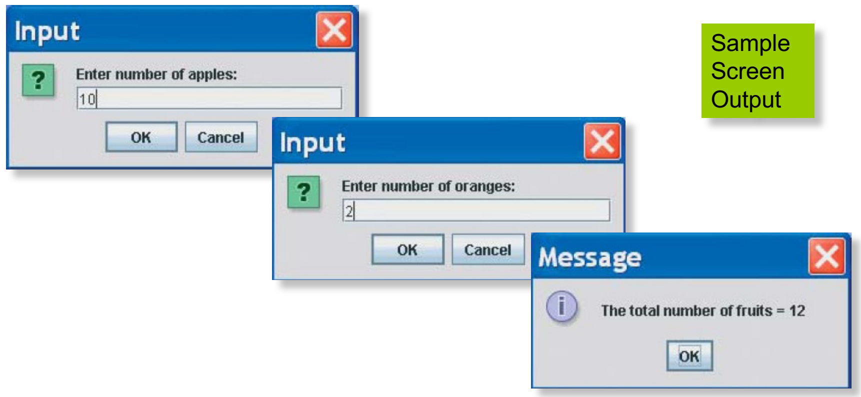
- **x, y and z need to be `double` and not `int`**

Debugging

- Many times it is very difficult to find the bug(s).
- You need to examine the code very carefully.
- First step
  - Suppose you have different inputs, then follow the code and determine the expected outputs.
- Second step
  - Run the code, and print the results in each step. See if the program matches with your first step.

GUI – `JOptionPane`

- An example of GUI – Graphic User Interface



Input
Enter number of apples:
10
OK Cancel

Input
Enter number of oranges:
2
OK Cancel

Message
The total number of fruits = 12
OK

Sample Screen Output

simple GUI

`import javax.swing.JoptionPane`

- Input

  ```
  String_Variable = JOptionPane.showInputDialog
     (String_Expression);
  ```
- Output

  ```
  JOptionPane.showMessageDialog
  (null, String_Expression);
  ```
- The the input and output is **ALWAYS** a String. So you need to convert (ex.: Integer.parseInt)
- `System.exit(0)` ends the program.

Example - SMSTio

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

        int count;
        double cost;

        System.out.print(“Input No of Messages: “);
        count = console.nextInt();
        cost = 10 + 0.22 * count;
        System.out.print(“Total Cost is “+cost);
   }
}
```

input

output

Example - SMSGui

input

```
import javax.swing.*;

public class SMSGui
{
  public static void main (String[] args)
  {
         int count;
         double cost;
         String str;

         count = Integer.parseInt(JOptionPane.showInputDialog("Input no of
                          messages: "));
         cost = 10 + 0.22 * count;
         str = "total cost = "+cost;
         JOptionPane.showMessageDialog(null, str, "SMS cost",
                         JOptionPane.INFORMATION_MESSAGE);
   }
}
```

output

TIO and GUI

- Try to transform TIO to GUI in at least one of the examples in the computer lab this week.
- Ask demonstrators for help

Example - triangle

- A triangle has the following properties:
  - No side of a triangle can be greater than the sum of the other two sides.
  - An isosceles triangle has two equal sides.
  - An equilateral triangle has three equal sides.
  - An scalene triangle has three different sides.
- Write a java code that read the three sides of a triangle and check if is a triangle and which type is.
- Try different pseudocodes/Java codes and try to compare them.

Example - salary

- Work out in a pseudocode and Java code to calculate the salary in the end of two weeks. You know that:
  - The salary rate is A$10/hour in the first 40 hours/week and A$15/hour after 40 hours/week.
  - If the total salary is less than 500, the worker will receive 10% bonus. If the salary is between 500 and 1000, the worker will receive 5% bonus.
- You will do this exercise during your next computer lab

Your task

- Read
  - Chapter 3 of the text book

☺

Have fun!!!

- Explore the Java API documentation
  - http://docs.oracle.com/javase/8/docs/api/
- Exercises
  - MyProgrammingLab
  - Implement/compile/run the examples from lecture slides
  - Complete the lab exercises