# SENG1110/SENG6110
# Object Oriented Programming

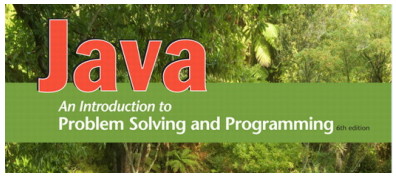
THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

www.newcastle.edu.au

Lecture 4

Control statements - loops




## Outline



- Previously…
  - 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
- Now…
  - Review – **if** and **switch**
  - Loop statements
    - **while**
    - **for**
    - **do…while**
    - **break** and **continue**
  - Examples
  - Testing



## Review - `if` and `if-else` - syntax



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



## Review - Relational and logical operators



- The following operators are used in relational statements:

```
<     // less than
<=    // less than or equal to
>     // greater than
>=    // greater than or equal to
==    // equal to
!=    // not equal to
&&    // AND
||    // OR
!     // NOT
```

## Review – 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 program that takes input for the three sides of a triangle, then checks if it is a triangle and which type is.
- Try different pseudocodes/Java codes and try to compare them.



## Review – Example – triangle



```
input s1,s2,s3;

if (s1<s2+s3 && s2<s3+s1 && s3<s1+s2)
   if (s1==s2 && s2==s3)
        output “Triangle is equilateral”
   else if (s1!=s2 && s1!=s3 && s2!=s3)
              output “Triangle is scalene”
         else
              output “Triangle is isosceles”
else
   System.out.println “It is not a triangle”
```



## Tio Template – try to write the code



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




  }
}
```



## Review - 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

# Review – Example – babies



```
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;
}
```





# Example - SMS cost pseudocode



```
input number_messages
cost = 10 + 0.22*number_messages
Output cost
```





# 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);
  }
}
```





# Example – SMS1



- Suppose now that you would like to calculate the same value for 12 different months.
- Loops
- Control statement

**while**
**do-while**
**for**

# `while` - syntax



- The while statement executes a loop
  - this executes the set of statements inside the loop whilst the condition remains true
  - if the condition is initially false, the statements inside the loop will not execute

```
while(<Boolean expression>)
     <statement>

while(<Boolean expression>)
{
     <statement 1>
     ...
     <statement n>
}
```





# Example – SMS1



- **Cost for 1 month**

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

- Cost for 12 different months

```
counter=1;
while(counter<=12)
{
input number_messages;
cost = 10 + 0.22*number_messages
output Totalcost
counter++;
}
```





# Example – SMS1



```
import java.util.*;
public class SMScost
{
    public static void main (String[] args)
    {
        Scanner console = new Scanner(System.in);
        int    count,counter=1;
        double cost;
        while(counter<=12)
        {
                System.out.print("Input No of Messages: ");
                count = console.nextInt();
                cost = 10 + 0.22 * count;
                System.out.println("Total Cost is "+cost);
                counter++;
        }
    }
}
```





# Loops controlled by a *Sentinel*



- Used when the exact number of entry pieces is unknown but the last entry (special/sentinel value) is known

```
input first data item into variable;
while(variable != sentinel)
{
      statement(s);
      input a data item into variable;
}
```

## Loops controlled by a *sentinel* - example



```
char answer = 'y'
while (answer == 'y')
{
    statements;
    output message "Do you want to continue?"
    input answer
}
```

- In this case, the user needs to enter with character 'y' if he/she wants to continue. If the user enters with any character different of 'y' the loop will stop.



## Loops controlled by a *counter*



- Used when exact number of input is known

```
int N = 5; //value input by user or specified in program
int counter = 0;
while(counter < N)
{
        System.out.print(i + " ");
        counter++;
}
```

**Output:** 0 1 2 3 4



## Loops controlled by a *counter* - example



```
int counter = 10;
while (counter > 0)
{
        System.out.println("counter = "+counter);
        counter = counter – 2;
}
```

**Output:** 10 8 6 4 2



## Useful operators - reminder



```
int a=9, b=5;
```

| | | |
|---|---|---|
| `a += 3;` | 12 | // Equivalent to a = a + 3; |
| `a -= 3;` | 6 | // Equivalent to a = a – 3; |
| `a *= 3;` | 27 | // Equivalent to a = a * 3; |
| `a /= 3;` | 3 | // Equivalent to a = a / 3; |
| `a++;` | 10 | // Equivalent to a = a +1; |
| `a--;` | 8 | // Equivalent to a = a - 1; |

## Loops controlled by a *counter* - example



- Write the numbers from 1 to 100

```
int counter = 1;
while (counter <= 100)
{

System.out.println(counter);
     counter++;
}
```

- Be careful with:
  - how you initialize the counter
  - defining your stop criteria
- Let's see different examples…





## Loops controlled by a *counter* - example



- Write the numbers from 1 to 100

```
int counter = 1;
while (counter <= 100)
{
  System.out.println(counter);
   System.out.println(counter);
  counter++;
}
```

Write -> 1…100
counter -> 1..101

```
int counter = 1;
while (counter < 100)
{
   System.out.println(counter);
   counter++;
}
```

Write -> 1…99
counter -> 1..100

```
int counter = 0;
while (counter <= 100)
{
  counter++;
}
```

Write -> 0…100
counter -> 0..101

```
int counter = 1;
while (counter <= 100)
{
  counter++;
  System.out.println(counter);
   System.out.println(counter);
```

Write -> 2…101
counter -> 1..101

```
int counter = 1;
while (counter < 100)
{
  counter++;
   System.out.println(counter);
```

Write -> 2…100
counter -> 1..100

```
int counter = 0;
while (counter <= 100)
{
  counter++;
```

Write -> 1…101
counter -> 0..101





## Loops controlled by a *counter* - example



- Write the numbers from 1 to 100, but only the odd ones.

```
int counter = 1;
while (counter <= 100)
{
     System.out.println (counter);
     counter+=2; //or counter=counter+2;
}
```





## Loops controlled by a *counter* - example



- Write the numbers from 100 to 1.

```
int counter = 100;
while (counter > 0)
{
     System.out.println(counter);
     counter--;
}
```

## Loops controlled by a *counter* - example



- Write the numbers from 100 to 5 that are multiples of 5.

```
int counter = 100;
while (counter >=5 )
{
     System.out.println(counter);
     counter-=5;
}
```





## Loops controlled by a *counter* - example



- Suppose now you want to write 10 numbers that the user inputs.

```
int counter = 1;
while (counter <= 10)
{
     number = console.nextInt();
     System.out.println(number);
     counter++;
}
```





## Loops controlled by a *flag*



- boolean value used to control loop

```
boolean found = false;
while(!found)
{
     statement(s);
     if(expression)
          found = true;
}
```





## Loops controlled by a *flag* - example



- boolean value used to control loop

```
final int number=8;
int n;
boolean found = false;
while(!found)
{
     System.out.print("Guess the number:");
     n = console.nextInt();
     if(n==number)
            found = true;
}
```

# Loops for accumulation - *sum*



- A common usage of while loops for accumulation

```
initialise an accumulator for the result
initialise a counter
while (some condition about the counter is true)
{
    perform a calculation;
    store the result in the accumulator;
    increment the counter;
}
```





# Loops for accumulation – *sum* - example



- Suppose now you want to sum the numbers from 1 to 100

```
int counter = 1;
int sum = 0;

while (counter <= 100)
{
     sum = sum + counter;
     counter++;
}

System.out.println(sum);
```

- Notice that here you need to write the answer only after the loop finishes. Why?





# Loops for accumulation – *sum* - example



- Suppose now you want to sum 100 numbers that the user inputs

```
int counter = 1;
int sum = 0;

while (counter <= 100)
{
     number =console.nextInt();
     sum = sum + number;
     counter++;
}

System.out.println(sum);
```





# Loops for accumulation – *sum* - example



- Sum numbers from **a** to **b**

```
int counter;
int sum = 0;
a = console.nextInt();
b = console.nextInt();
counter = a;

while (counter <= b)
{
     sum = sum + counter;
     counter++;
}

System.out.println(sum);
```

But, if a>b?

## Loops for accumulation – *sum* - example



- Sum numbers from **a** to **b**

```
int counter,sum=0,aux;
a = console.nextInt();
b = console.nextInt();
if (a>b)
{
    aux=a;
    a=b;
    b=aux;
}
counter = a;
while (counter <= b)
{
    sum = sum + counter;
    counter++;
}
System.out.println(sum);
```



## Loops for accumulation – *prod* - example



- 1*2*…*100

```
int counter = 1;
int prod = 1;

while (counter <= 100)
{
    prod = prod * counter;
    counter++;
}

System.out.println(prod);
```



## Loops for accumulation – *prod* - example



- n! = 1*2*3*…*n

```
int counter = 1;
int prod = 1;
n = console.nextInt();

while (counter <= n)
{
    prod = prod * counter;
    counter++;
}

System.out.println(prod);
```



## Loops for accumulation – *prod* - example



- $2^n$

```
int counter = 1;
int prod = 1;
n = console.nextInt();

while (counter <= n)
{
    prod = prod * 2;
    counter++;
}

System.out.println(prod);
```

## Designing Correct Loops



- Initialize all variables properly
  - Using a counter, plan the number of iterations, then set the counter and the limit accordingly
- Check the logic of the termination condition
- Update the loop control variable properly





## Example – SMS2



- SMS example again…
  - Suppose we want to calculate the total cost for 12 different months

    and

  - in the end to give the average cost during the year. How to do this?





## Example – SMS2



- **Calculate just one total cost**

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

- **Calculate total cost**

```
counter=1;
while(counter<=12)
{
    input number_messages;
    cost = 10 + 0.22*number_messages
    output Totalcost
    counter++;
}
```





## Example – SMS2



- **Calculate total cost and the average in 12 months**

```
counter=1
sum=0

while(counter<=12)
{
    input number_messages
    cost = 10 + 0.22*number_messages
    sum = sum + cost
    output cost
    counter++;
}

average=sum/12
output average
```

# Example – SMS2



```
import java.util.*;
public class SMScost
{
    public static void main (String[] args)
    {
        Scanner console = new Scanner(System.in);
        int    count,counter=1;
        double cost,average=0;

        while(counter<=12)
        {
            System.out.print("Input No of Messages: ");
            count = console.nextInt();
            cost = 10 + 0.22 * count;
            System.out.println("Total Cost is "+cost);
            counter++;
            average+=cost;
        }
       average/=12;
       System.out.println("Avegare Cost is "+average);
   }
}
```





# Example – SMS3



- SMS example again…
  - Suppose you want to calculate the total cost for M different months

    and

  - in the end to give the average cost during the period. How to do this?





# Example – SMS3



- **Calculate total cost and the average in M months**

```
counter=1
sum=0
input M

while(counter<=M)
{
    input number_messages
    cost = 10 + 0.22*number_messages
    sum = sum + cost
    output cost
    counter++
}

average=sum/M
output average
```





# Example – SMS3



```
import java.util.*;
public class SMScost
{
    public static void main (String[] args)
    {
        Scanner console = new Scanner(System.in);
        int    count,counter=1,M;
        double cost,average=0;

        while(counter<=M)
        {
            System.out.print("Input No of Messages: ");
            count = console.nextInt();
            cost = 10 + 0.22 * count;
            System.out.println("Total Cost is "+cost);
            counter++;
            average+=cost;
        }
       average/=M;
       System.out.println("Average Cost is "+average);
   }
}
```

## do...while statement



```
do
{
   statement(s);
}
while(expression);
```

- Statements executed first, then expression evaluated
- Statement(s) executed at least once then continued if expression is true




## do...while statement - example



```
i = 0;

do
{
    System.out.print(i + " ");
    i = i + 5;

}
while(i <= 20);
```

**Output:** 0 5 10 15 20




## The for statement



```
for(initial statement; loop condition; update statement)
{
   statement(s);
}
```

- Execution:
  - `initial statement` executes
  - `loop condition` evaluated
  - If the `loop condition` evaluates to true, execute for loop statement and execute `update statement`
  - Repeat until `loop condition` is false




## The for statement



- Does not execute if initial condition is false
- Update expression changes value of loop control variable, eventually making it false
- If loop condition is always true, result is an infinite loop
- Infinite loop can be specified by omitting all three control statements `for (;;)`
- If loop condition is omitted, it is assumed to be true



## The `for` statement - example



```
for(int i = 10; i >= 1; i--)
    System.out.print(" " + i);
```

Output: 10 9 8 7 6 5 4 3 2 1

```
for(int i = 1; i <= 20; i = i + 2)
    System.out.print(" " + i);
```

Output: 1 3 5 7 9 11 13 15 17 19




## Example – Classify Numbers



- Input:
  - N integers (positive, negative, and zeros)
    int N = 20; //N easily modified
- Output:
  - number of 0s, number of even integers, number of odd integers




## Example – Classify Numbers



- Use a `for` loop to get and evaluate 20 numbers from the user and input into switch structure
- `switch`(number % 2)
- case 0:
  - if(number == 0) increment number of zeros;
  - if(number != 0) increment number of even integers;
- case 1:
  - increment number of odd integers




## Example – Classify Numbers



```
import java.util.*;

public class ClassifyNumbers
{

   static final int N = 20;

   public static void main(String[] args)
   {
         Scanner console = new Scanner(System.in);
         int counter;    //loop control variable
         int number;     //variable to store the new number
         int zeros = 0,odds = 0, evens = 0;

         System.out.println("Please enter " + N + " integers, "
                     + "positive, negative, or zeros.");

 //continued on next slide
```



## Example – Classify Numbers



```
for(counter = 1; counter <= N; counter++)
{
     number = console.nextInt();
          System.out.print(number + " ");
     switch(number%2)
          {
            case 0:  evens++;
                     if(number == 0) zeros++;
                     break;
            case 1:  odds++;
      } //end switch
  } //end for loop

 System.out.println("\nThere are " + evens + " evens\n"
 + "which also includes " + zeros + " zeros\n"
                        + "Total number of odds is: " + odds);
  }
}
```




## `break` statement



- Used to exit early from a loop
- Used to skip remainder of switch structure
- Can be placed within if statement of a loop
  - If condition is met, exit the loop immediately




## `continue` statement



- Used in `while`, `for`, and `do...while` structures
- When executed in a loop, the remaining statements in the loop are skipped; proceeds with the next iteration of the loop
- When executed in a while/do…while structure, the logic expression evaluated immediately after continue statement
- In a for structure, the update statement is executed after the continue statement; then the loop condition executes




## `break` and `continue` examples



```
for(i=1;i<10;i++)
{
    if (i%3==0) continue;
    System.out.print(i);
}
```

- It will display 124578

```
for(i=1;i<10;i++)
{
    if (i%3==0) break;
    System.out.print(i);
}
```

- It will display 12



# Nested Control Structures



- Provides new power and complexity
- if, if…else, and switch structures can be placed within loops
- loops can be found within other loops





# Nested Control Structures (Example)



```
for(int i = 1; i <= 5; i++)
{
    for(int j = 1; j <= i; j++)
    System.out.print("*");
    System.out.println();
}
```

- Output:

```
*
**
***
****
*****
```





# Nested Loops - Example 5 - ExamAverage

```
Want to average another exam?
Enter yes or no.
yes

Enter all the scores to be averaged.
Enter a negative number after
you have entered all the scores.
90
70
80
-1
The average is 80.0
Want to average another exam?
Enter yes or no.
no
```



# Example – ExamAverage



```
import java.util.Scanner;

/** Computes the average of a list of (nonnegative) exam scores. Repeats
computation for more exams until the user says to stop. */

public class ExamAverager
{
   public static void main (String [] args)
   {
      Scanner keyboard = new Scanner (System.in);
      double sum, next;
      int numberOfStudents;
      String answer;

      System.out.println ("This program computes the average of");
      System.out.println ("a list of (nonnegative) exam scores.");
```

//continued on next slide

```
    do
    {
        System.out.println ("Enter all the scores to be averaged.");
        System.out.println ("Enter a negative number after");
        System.out.println ("you have entered all the scores.");
        sum = 0;
        numberOfStudents = 0;
        next = keyboard.nextDouble ();
        while (next >= 0)
        {
           sum = sum + next;
           numberOfStudents++;
           next = keyboard.nextDouble ();
        }
        if (numberOfStudents > 0)
           System.out.println ("The average is " + (sum / numberOfStudents));
        else
           System.out.println ("No scores to average.");

        System.out.println ("Want to average another exam?");
        System.out.println ("Enter yes or no.");
        answer = keyboard.next ();
    }
    while (answer.equalsIgnoreCase("yes"));
  }
}
```



## Example – Repayments



- Write a program that calculates the repayment/month when you buy a product which will be paid off in M months with an interest calculated monthly.
- Inform the user how much the total interest is at the end of the period



## Example – Repayments



- Analysis results in:
  - The inputs are
    - the product price
    - the interest rate
    - the number of months
    - The initial payment (optional)
  - The outputs are
    - the repayment/month
    - the total interest



## Example – Repayments



- Output

```
No of Months? 2
Rate? 10 (means 10%)
Product price: 1500
Initial Pay? 100

Repay in the month 1 = 770.0
Repay in the month 2 = 847.0
Total pay = 1617.0
Total interest = 217.0
```

1500-100 = 1400
1400/2 = 700
1 -> 700*1.1 = 770
2 -> 770*1.1 = 847

## Example – pseudocode



```
input months
input rate/100
input price
input initial_pay

price-=initialpay;
repay = price/months;
totalpay=0;

while (months > 0)
{
    repay=repay+repay*rate
    totalpay += repay
    output repay
    months = months - 1
}

output totalpay
output (totalpay-price)
```



## Tio Template – try do write the code



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



  }
}
```



## Testing – conditional and loop statements



- It is important to test whether your program works correctly
- This involves
  - testing whether the program executes without crashing

  AND

  - testing whether the program does what it is meant to do



## Testing - if



- The testing should thus
  - verify the logic of the Boolean expressions
  - verify that the symbols have been used the correct way around
    - eg "<" or ">"  ???
- Always verify if there is another way (best) to build a nested if

## Testing - if



- You should choose test data to exercise all alternative paths through the selection statements

```
if (count > 50)
   <compute discount>
else if (count >= 0)
   <compute std>
else
   <display error message>
```

| Value of count | Expected Result |
|---|---|
| 10 | compute std |
| 73 | compute discount |
| -3 | error message |

- Obvious places to test are so-called boundary conditions, in this case -1, 0 & 1 and 49, 50 & 51.
- Why are these boundary conditions?



## Loop Errors - 1



- Typically a loop has four components
  - initialising statements
    - set up variables used in the loop
  - terminating condition
    - determines whether another iteration occurs
  - body statements
    - execute each iteration
  - update statements
    - change value(s) tested in terminating condition



## Loop Errors - 2



- Errors can occur in any of these parts. Consider the following correct code:

```
// Compute the product of the odd integers from
// 1 to 100 (ie 1*3*5* … *99)

product = 1;
currentNumber = 3;
while (currentNumber <= 100)
{
    product = product * currentNumber;
    currentNumber = currentNumber + 2;
}
```



## Loop Errors - 3



- An initialisation error

```
currentNumber = 3;
while (currentNumber <= 100)
{
    product = product * currentNumber;
    currentNumber = currentNumber + 2;
}
```

- Fails to initialise the variable 'product'
  Compiler detects problem

## Loop Errors - 4



- Another initialisation error

```
product = 0;
currentNumber = 3;
while (currentNumber <= 100)
{
    product = product * currentNumber;
    currentNumber = currentNumber + 2;
}
```

- Initialises 'product' to zero
- Result will be zero




## Loop Errors - 5



- Error in the terminating condition

```
product = 1;
currentNumber = 3;
while (currentNumber < 99)
{
    product = product * currentNumber;
    currentNumber = currentNumber + 2;
}
```

- Uses '<99' instead of '<100'
- Product equals 1*3*5* … *97




## Loop Errors - 6



- Called an "off-by-one" error. Compare

```
int counter = 1;
while (counter <= 10)
{
   <do something>
   counter++;
}
```

// Executes 10 passes

```
int counter = 1;
while (counter < 10)
{
   <do something>
   counter++;
}
```

// Executes 9 passes




## Loop Errors - 7



- Another error in the terminating condition

```
product = 1;
currentNumber = 3;
while (currentNumber != 100)
{
    product = product * currentNumber;
    currentNumber = currentNumber + 2;
}
```

- Uses '!=' instead of '<='
- Program never stops … an *infinite* loop



## Loop Errors - 8



- Called an infinite loop. Compare

```
int counter = 1;
while (counter <= 10)
{
   <do something>
   counter = counter + 2;
}
```

Executes 5 passes

```
int counter = 1;
while (counter != 10)
{
   <do something>
   counter = counter + 2;
}
```

Runs forever





## Loop Errors - 9



- Error in the body

```
product = 1;
currentNumber = 3;
while (currentNumber <= 100)
{
    product = product + currentNumber;
    currentNumber = currentNumber + 2;
}
```

- Uses '+' instead of '*'
- Product incorrect





## Loop Errors - 10



- Error in the update statement

```
product = 1;
currentNumber = 3;
while (currentNumber <= 100)
{
    currentNumber = currentNumber + 2;
    product = product * currentNumber;
}
```

- Update statement misplaced
- Product calculated is 5*7*9* … *99*101





## Your task



- Read
  - Chapter 4 of the text book


- Exercises
  - MyProgrammingLab
  - Implement/compile/run the examples from lecture slides
  - Complete the lab exercises