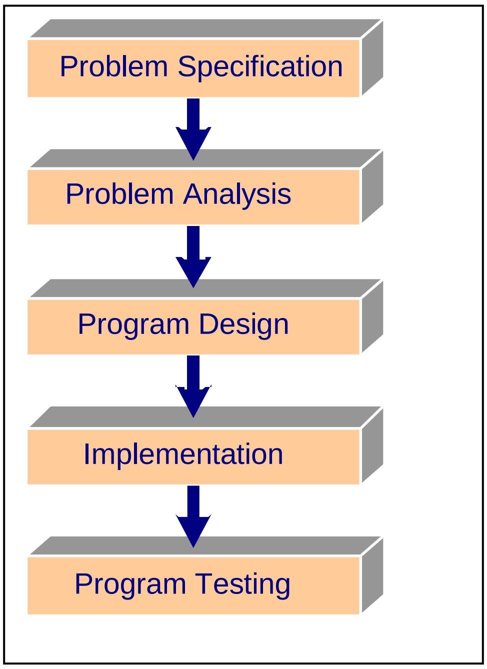
Software Development Process

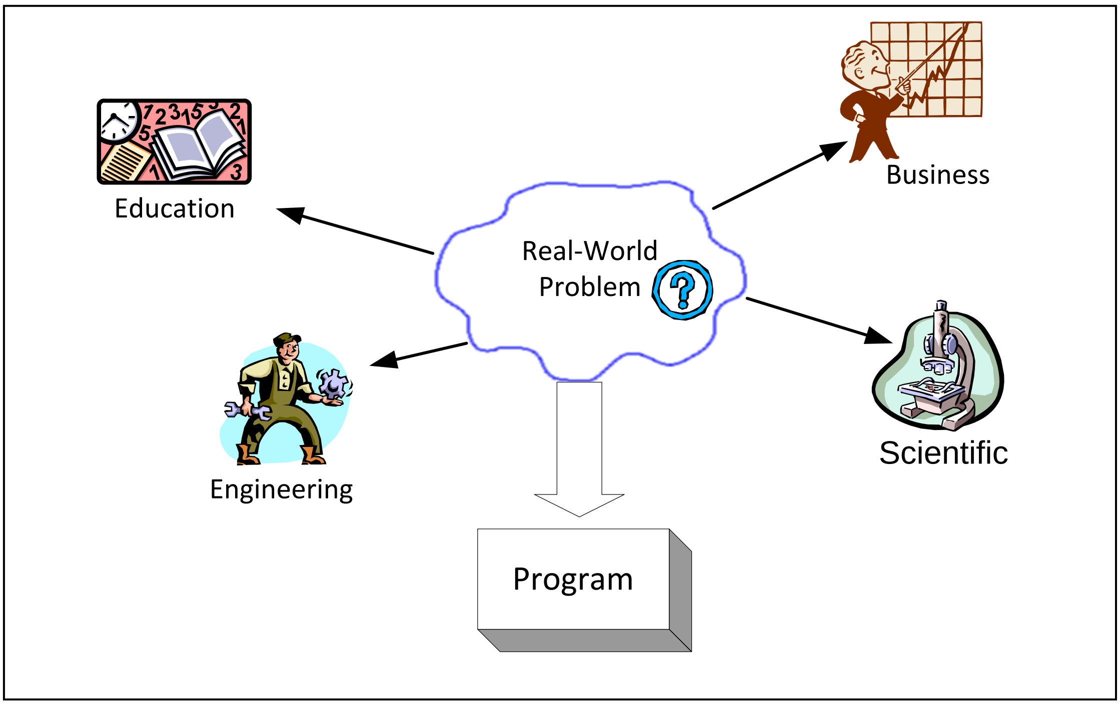
The software development process generally goes through 5 phases.



1) **Problem Specification**
   a. This is a broad statement of the user requirements, in user terms. It sets the program's goals the program's boundaries – the inputs and outputs
2) **Problem Analysis**
   a. This phase produces a set of clear statements about the way the program is to work. These statements define:
   how the user uses the software(input)
   what output the software will create after being triggered. The formulas (algorithm, mathematical computations) of the software.
   Additional requirements and or constraints discovered after this stage.
3) **Program Design**
   a. Design the logic of the software or algorithm to use.
   Analyse the problem. It must be verified with the data gathered at the system specification stage. The design of algorithms that gives the steps to transform the inputs into the intended outputs. Uses techniques such as flowchart or pseudo code.

   b. Find appropriate API (Application Programming Interface) to be used for problem solving. This could be very helpful for reducing workload or re-inventing the wheel.
4) **Implementation**
   a. The coding stage. Initial Algorithm can be formed from the Initialization of variables, Read the input data, Perform the computation and finally Trigger the outputs.
5) **Program Testing**
   a. The software are tried, tested and feed back by users. Bugs can be discovered during UAT (User Acceptance Test)

Programming is used to solve real world problems



Describe any two (2) real world challenges that could be solve using code, hardware. Draw a diagram on how your new system would help solve the problem in the space given below.

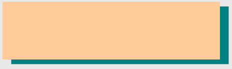
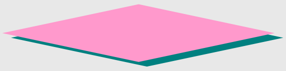
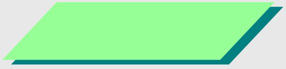
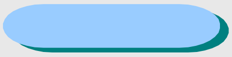
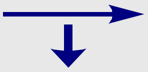
Designing an algorithm (e.g steps to solve a problem) in a program

Use English (noun + verb) to describe the action/logic of the program.

When designing an algorithm

1. The description must be unambiguous.
2. Each step into the problem solving must be clear and precise.
3. Specify the order of executing the steps precisely.
4. Consider all possible points of discourse during the decision making.
5. The algorithm must execute in steps.
6. The program will terminate in finite time.

The outcome of the design can be represented using Pseudocode or Flowcharts (preferred method)

Symbol	Name
	Process
	Decision
	Input / Output
	Terminal
	Flowlines

It is easier to visualise the flow of the control (the algorithm) and can be easily communicated to other people that are not technically inclined. This is a way to represent the logic or solution method of a program by diagram. Note the symbol to represent the different items.

A more formalized flowchart notation will be the UML Activity diagram. http://en.wikipedia.org/wiki/Activity_diagram

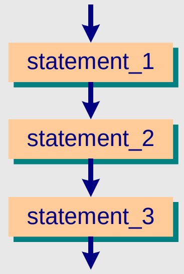
Typical types of programming errors

Syntax Errors
"grammatical" errors, detected by compiler, found automatically need to be fixed before running the code.

Logic Errors
Maybe due to error in designing the algorithm or implementation. Difficult to detect.

Runtime Errors
detected during the execution of program

Pseudocode	Flowchart

begin
    statement_1;
    statement_2;
    statement_3;
end
	

There are no strict rules regarding pseudo code. There are programming language that looks similar to pseudo code, such as **python**.

Pseudocode is very informal type language using a mix of English and common "programming" keywords such as begin, end, if, else, else_if, while, do, do_while, break, read/write, display/print, initilaize, compute,and etc.

This can be quote mindboggling for somebody that is not used to writing/reading this type of text.

Documentation is needed for further modification and maintenance.
Proper documentation includes:
problem definition and specification;
program inputs, outputs, constraints and mathematical equations;
flowchart or pseudo code for the algorithm; source code listing;
sample test run of the program; and
user manual for end users.

Exercise (**BASIC**) : Design steps to solve a classic problem.

Procedure: Draw the flow chart on the processes/steps of the calculation(algorithm) and write using pseudo code in the space provided below.

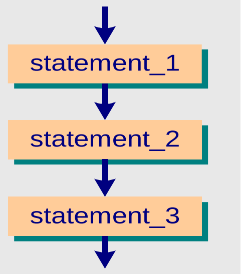
1. Software that converts local currency into Euro.

2. Design an algorithm using flowchart and pseudo code that create a conversion table to convert degrees Celsius to degree Fahrenheit. The user is to enter temperature in Celsius and display in Fahrenheit. A friendly prompt to remind the user the data input is celcius instead of Fahrenheit. (Fahrenheit = 32+ (9*Celcius/5))

Basic components of a program (procedural &&|| object oriented)

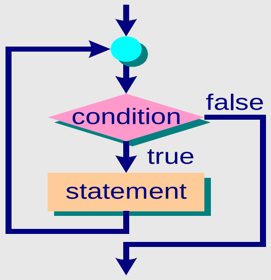
A simple program will use at least one of the 3 most common programming components, which is the **sequence** statements, **branching** statements and **looping** statements.

Pseudocode	Flowchart

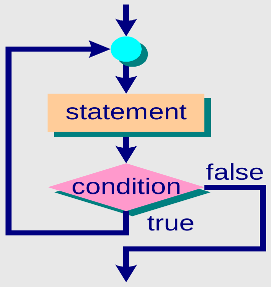
begin
statement_1;
statement_2;
statement_3;
end
	

Sequence Statements: Usually are method calls.

while structure	
Pseudocode	Flowchart

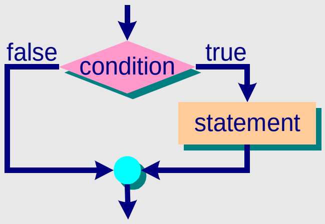
while (condition)
statement;
end_while
	

do-while structure	
Pseudocode	Flowchart

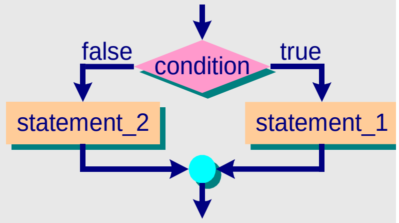
do
statement;
while (condition)
	

Looping statements: Usually are **While** statements, **Do-While** statements and **For** statements.

if structure	
Pseudocode	Flowchart

if (condition)
statement;
end_if
	

if-else structure	
Pseudocode	Flowchart

if (condition)
statement_1;
else
statement_2;
end_if
	

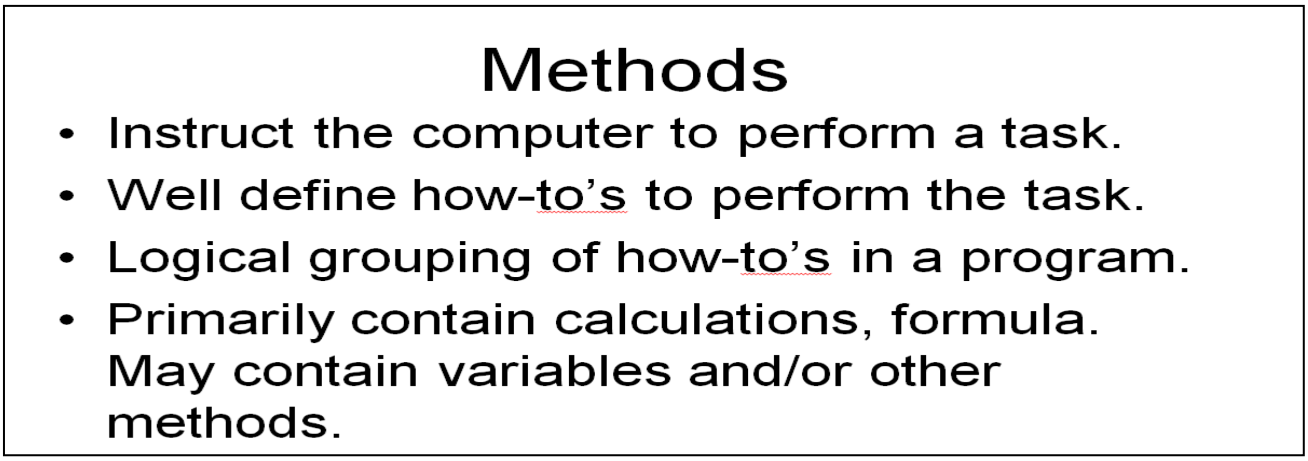
Branching statements: usually are If-Else statements. Take extra caution to differentiate between **nested** If statements and **cascaded** if statements. The prior would need all conditions to held through, the latter with just 1 condition.

Other branching statements such as **Switch** statements are possible too, and it can be used as an alternative to a myriad of cascaded If statements.

Program Testing

A program can be considered to be correct, if the results are expected from the combinations of inputs, or errors are properly handled. It is impractical to test for each and every possible inputs as the program complexity grew. Hence, each program path must be tested instead.

A Primer on Programming: Methods/Function Calls



Methods

- Instruct the computer to perform a task.
- Well define how-to’s to perform the task.
- Logical grouping of how-to’s in a program.
- Primarily contain calculations, formula. May contain variables and/or other methods.

may consist of the following. Take special note on the colour and the highlighted portion of the text. Programming is sensitive to syntax and formatting.

```
return_type method_name (parameter list){
//formula here
}
```

return type: variable/status to be returned to the calling function

method_name: a user define name of the method/function, use this created name to call the method.

Parameter list: the variables to be passed into the methods, using it to perform calculations.

Formula: referring to the calculations.

Example: Lecturer ask class rep to go and collect pizza.
boolean status = collectPizza(andy, 20.99);

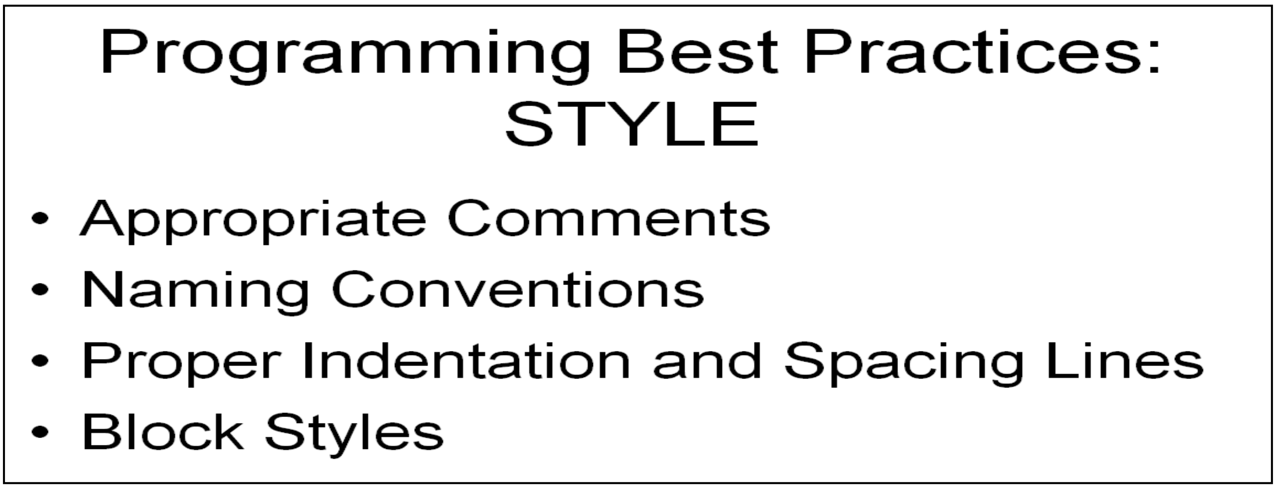
```
boolean collectPizza(student classRep, double money){

//pseudo code
classRep find out how to go to shop;
present order to cashier;
pay money;
collect result;
return success;

}
```

From the above code, identify the

1. Caller and called of the method/function.
2. Variables used, list down the name/amount.
3. Variable type used.
4. Purpose of return type in this example.

CHALLENGE Implement your code/algorithm in a language of your chouce, e.g python, java, c++, etc

A Primer on Programming: Programming style

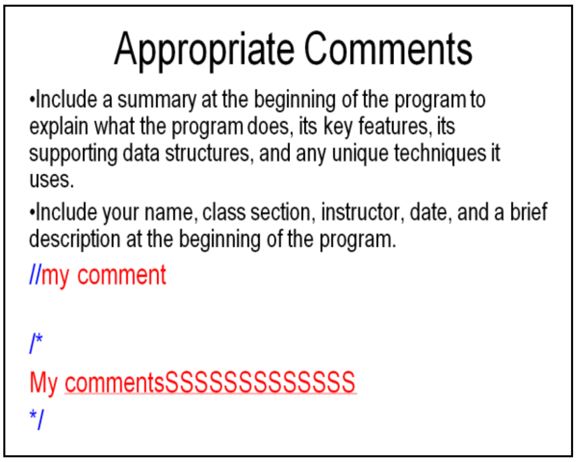


Programming Best Practices: STYLE

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles

Improve on the code readability by others, appropriate comments actually help a lot during the initial stage of looking at code.

Following the standard naming convention can help significantly when programming with many people. Indentation helps in troubleshooting and understanding code sections.

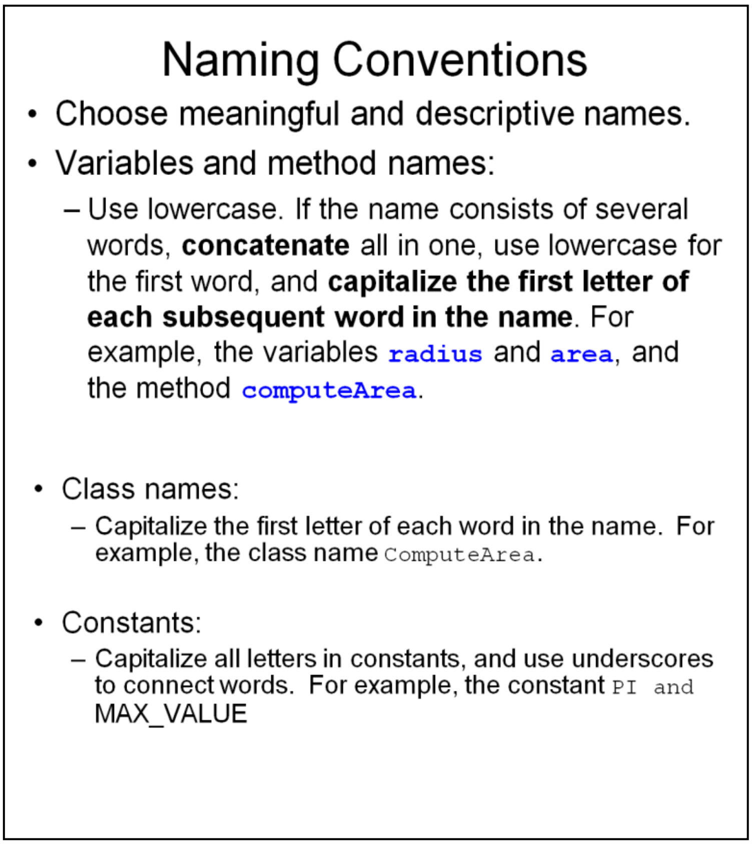


Appropriate Comments

•Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

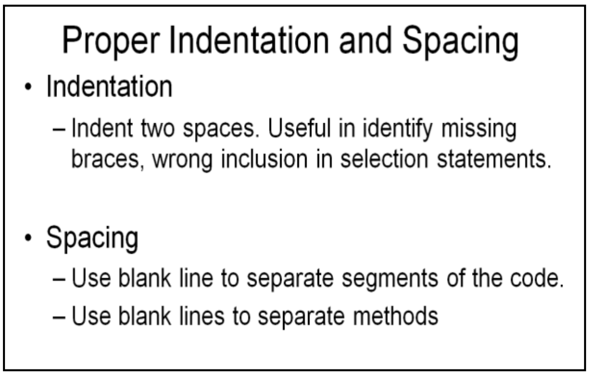
•Include your name, class section, instructor, date, and a brief description at the beginning of the program.

```
//my comment

/*
My commentsSSSSSSSSSSSS
*/
```



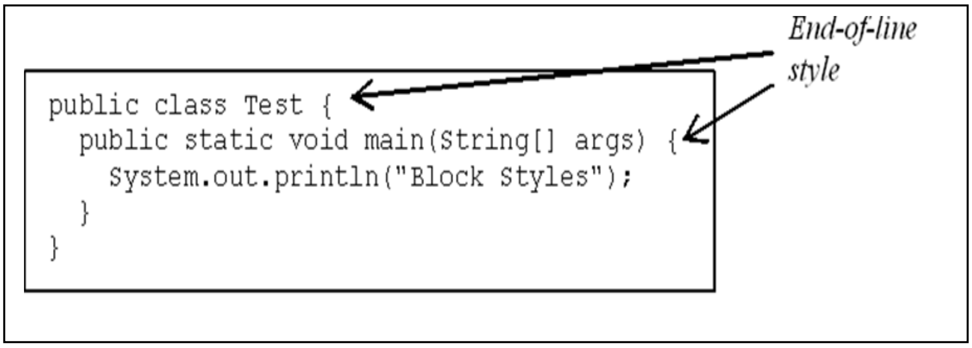
Naming Conventions

- Choose meaningful and descriptive names.
- Variables and method names:
  - Use lowercase. If the name consists of several words, **concatenate** all in one, use lowercase for the first word, and **capitalize the first letter of each subsequent word in the name**. For example, the variables `radius` and `area`, and the method `computeArea`.
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeArea`.
- Constants:
  - Capitalize all letters in constants, and use underscores to connect words. For example, the constant `PI and` MAX_VALUE



Proper Indentation and Spacing

- Indentation
  - Indent two spaces. Useful in identify missing braces, wrong inclusion in selection statements.
- Spacing
  - Use blank line to separate segments of the code.
  - Use blank lines to separate methods



End-of-line style

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```