Programming Foundation

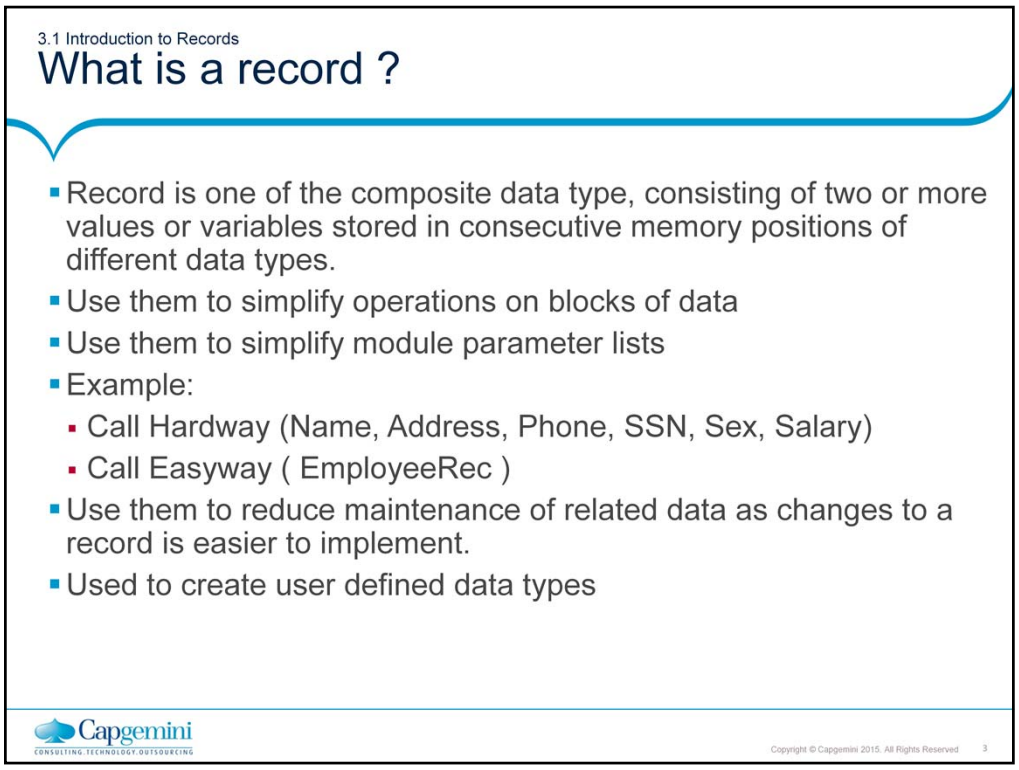
Lesson 3: File Handling & Refactoring

Lesson Objectives

- To understand the following concepts:
  - Records
  - File Handling
  - Re-factoring
  - Avoidance of common programming mistakes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

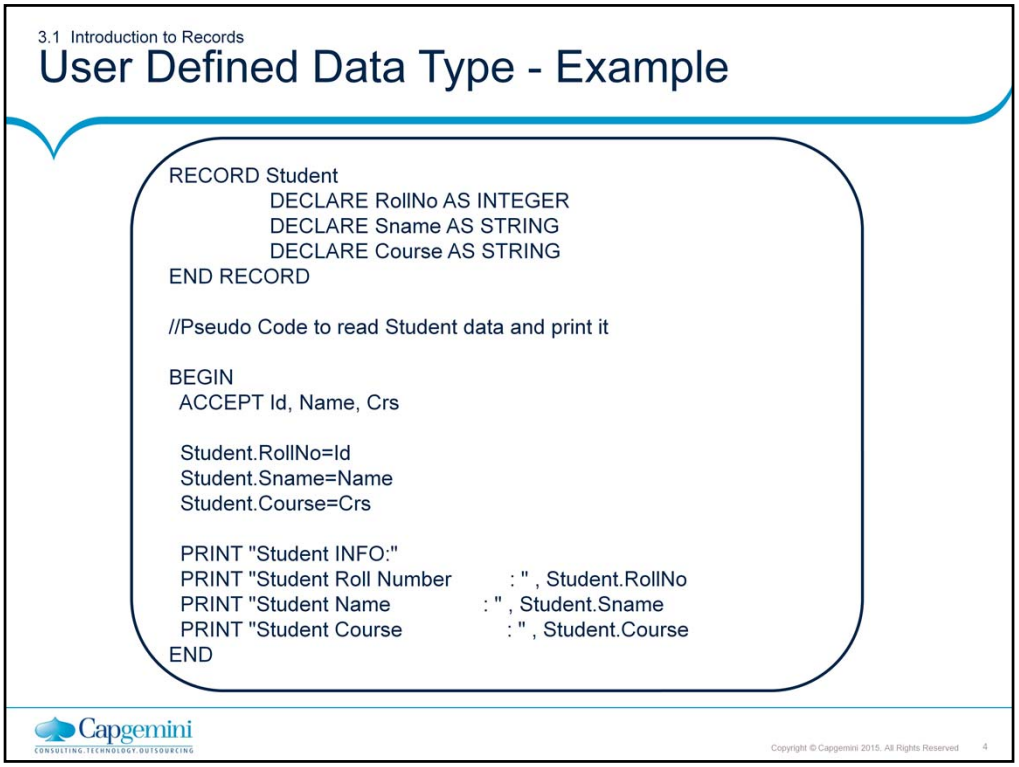


Record: Record is one of the composite data type, consisting of two or more values or variables stored in consecutive memory positions .

Example for record:

```
RECORD Employee
        DECLARE ecode AS INTEGER
        DECLARE ename AS STRING
        DECLARE esal AS INTEGER
        DECLARE edept AS STRING
END RECORD
```

The above record is used to hold details about an employee such as employee code, employee name, employee salary and department in which employee is working.

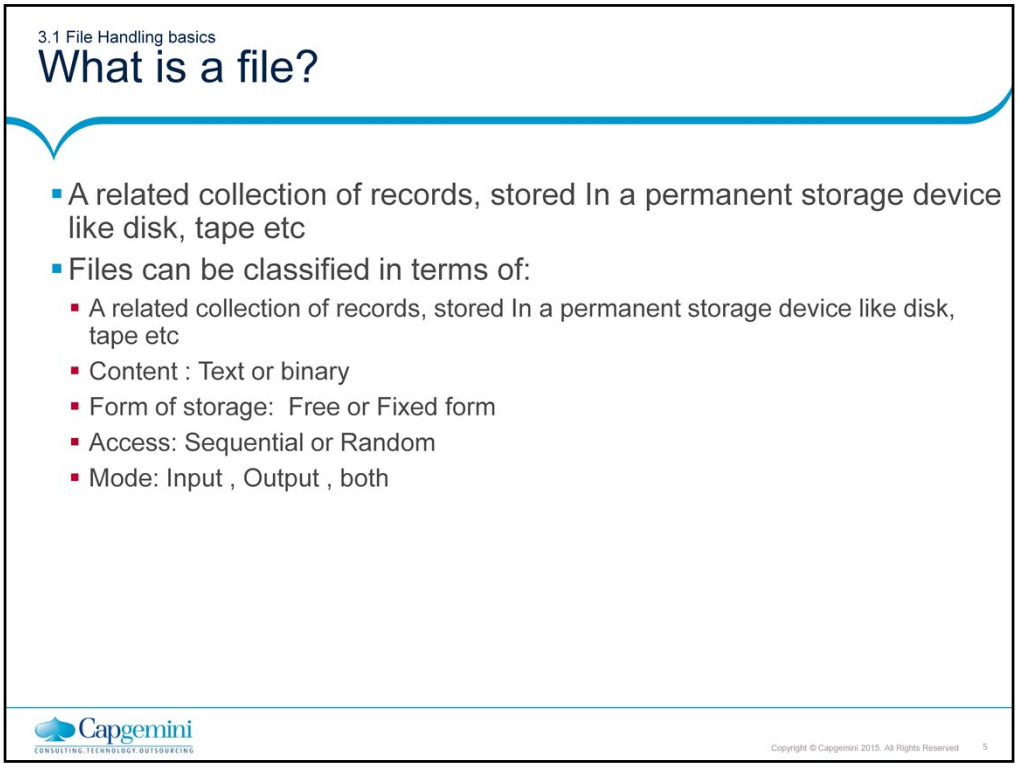


The code given in slide is used to maintain information about a student like rollno, sname and course details.

Functionalities implemented in the above code are

Accepting student details like id, name and course

Printing the same.



Files:

So far the input data was read from standard input and the output was displayed on the standard output. These programs are adequate if the volume of data involved is not large. However many business-related applications require that a large amount of data be read, processed, and saved for later use. In such a case, the data is stored on storage device, usually a disk in the form of file.

There are two types of files:

Binary file:

It is efficient for large amount of numeric data.

Text file:

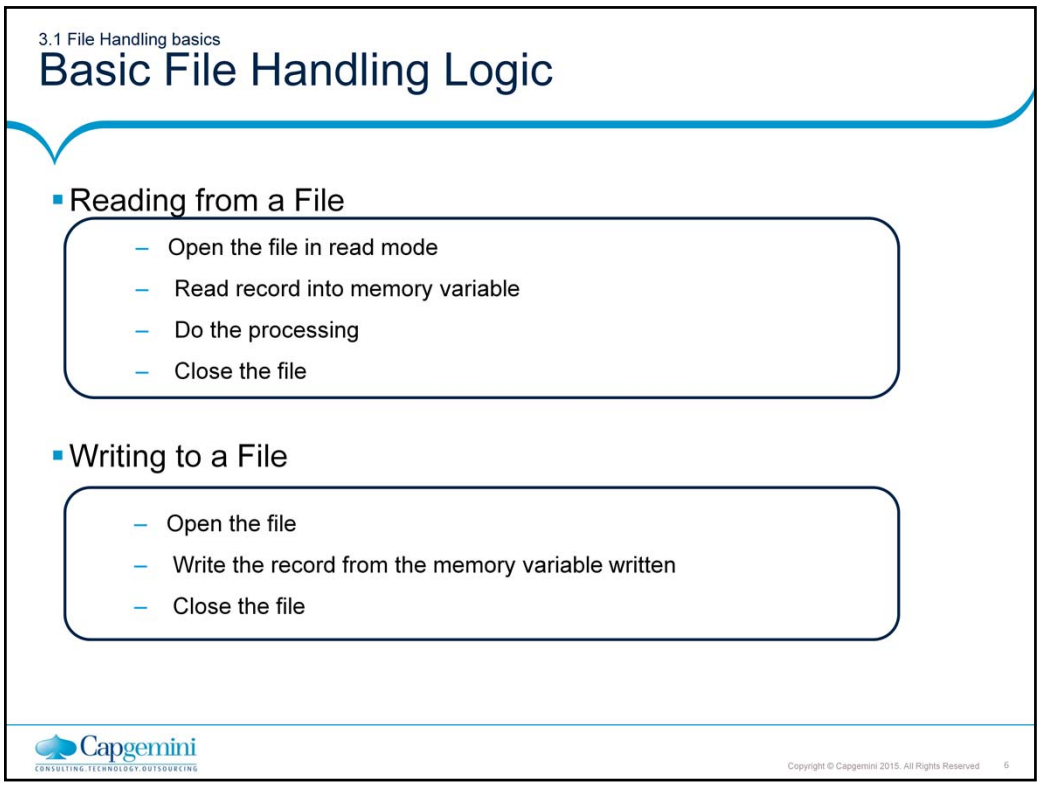
It stores text and numbers as one character per byte.

It consumes large amount of storage for numbers.

It is useful for printing reports and text documents.

Access: Data from the file is accessible either sequentially or randomly.

Mode: Data from the file can be readable(Output) or data can be writable to the file(Input).



File I/O requires four functions:

- **open:** It allows access to file.
- **close:** It ends access to a file.
- **read:** It gets data from file.
- **write:** It adds data into file.

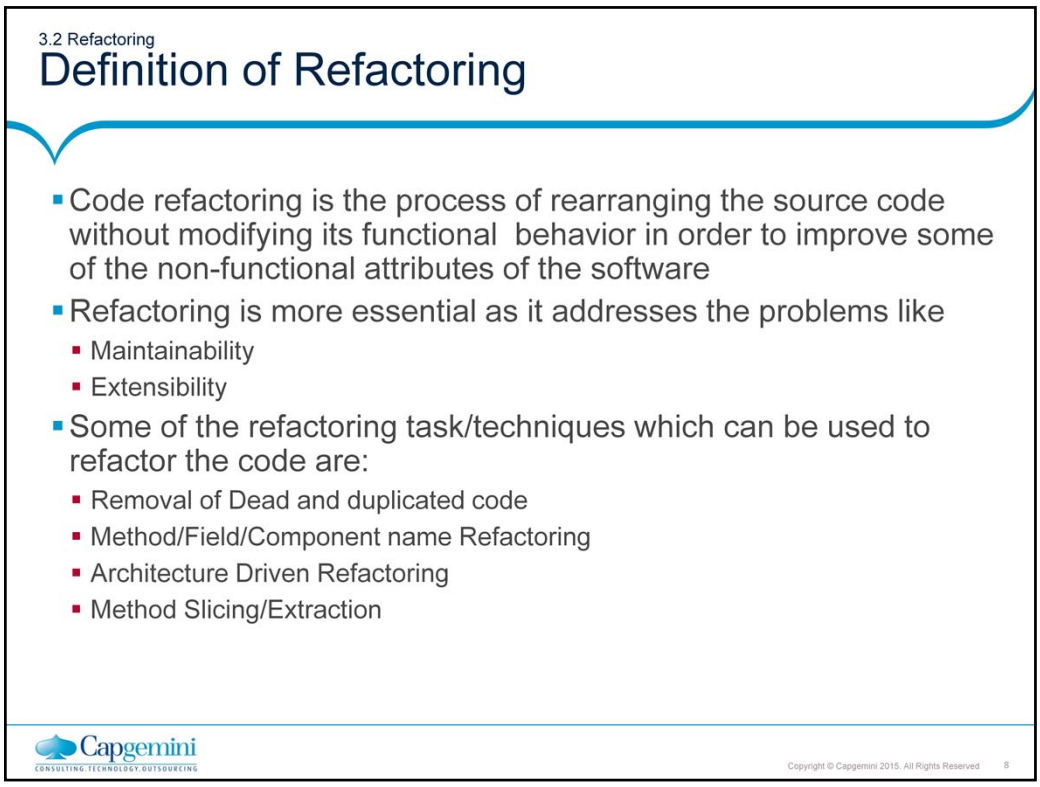
Demo : Reading data from file

- A product file contains the following fields
  - Product Id
  - Name
  - Price
- Refer the pseudo code in the notes page which will list all the products available in the file whose price is above 5,000

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

```
RECORD ProductRec
        DECLARE productId AS INTEGER
        DECLARE name AS STRING
        DECLARE price AS INTEGER
END RECORD
BEGIN
        DECLARE file AS FILE
        DECLARE product AS ProductRec
        PROMPT "Enter the filename" AND STORE IN file
        IF(fileExists(file)) THEN
                OPEN file
                READ data from the file AND STORE IN product
                IF(product.price>5000) THEN
                                DISPLAY "Product Id" + product.productId
                                 DISPLAY "Product Name" + product.name
                                 DISPLAY "Product Price" + product.price
                END IF
         END IF
END
```



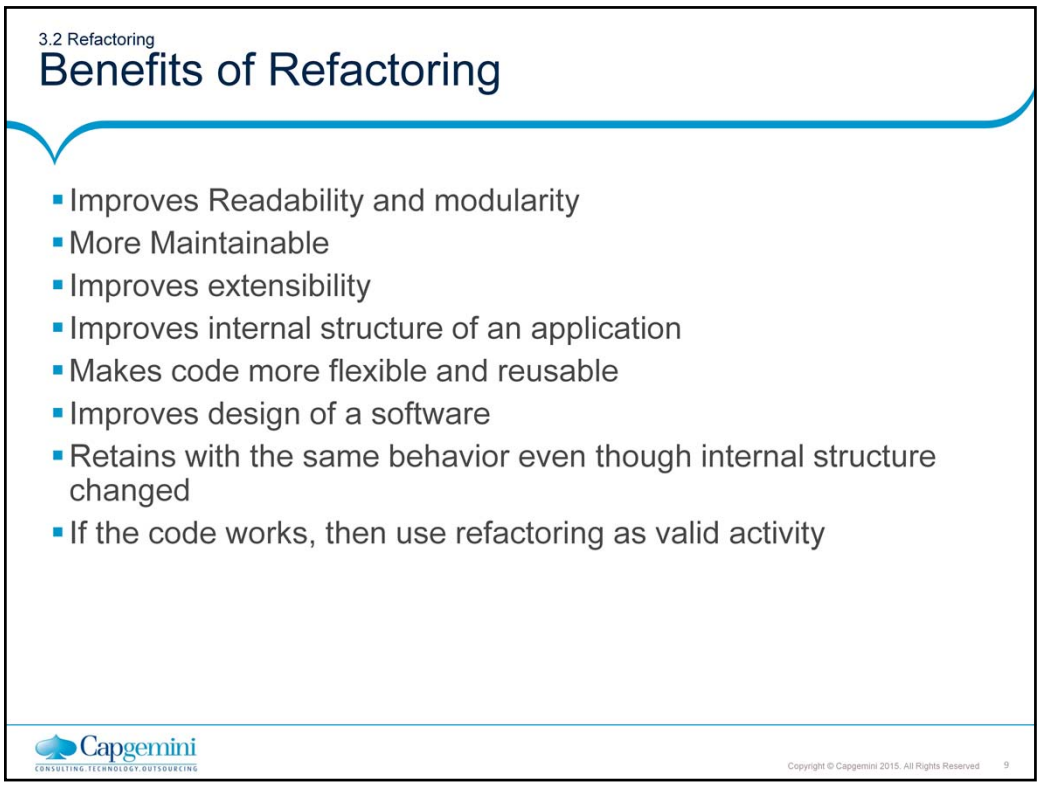
Refactoring is the process of clarifying and simplifying the design of existing code, without changing its behavior. Refactoring requires care because it can introduce bugs to the code

Refactoring is more essential as it addresses the problems like

- Maintainability – Code is not maintainable
- Extensibility – Very difficult to add new feature or upgrade an existing feature

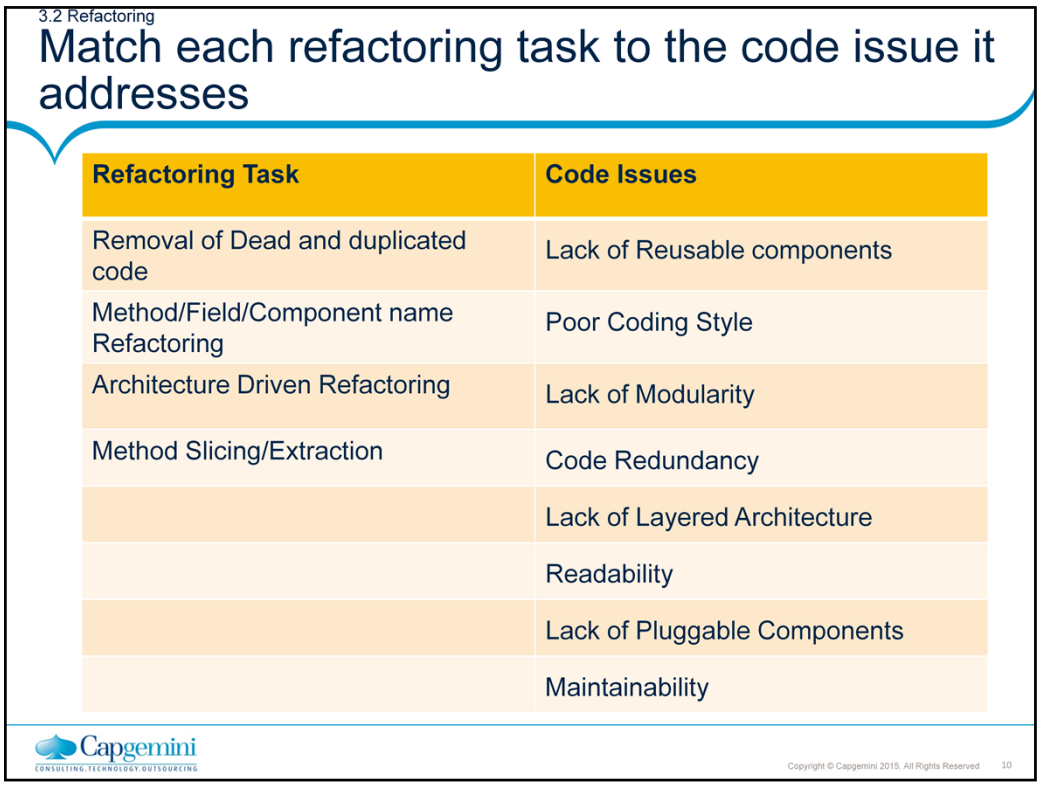
Refactoring task/techniques are:

- Removal of Dead and duplicated code - Remove the unused variables/code, unreachable statements and duplicated code.
- Method/Field/Component name Refactoring – Name of a method/field/component might be confusing, provide a meaningful name for the same and ensure the changes are reflected in other references wherever the variable/field/component is used.
- Architecture Driven Refactoring - Few functionalities in an application can be potentially reused in other applications. In order to reuse such a functionality, separate the code in a separate component by adhering to the architecture design.
- Method Slicing/Extraction - Longer method needs to be broken up into smaller ones to enhance readability and maintainability. Also need to achieve cohesion by implementing only one logic in a method with more suitable name.



Benefits of Refactoring:

- Improves code readability and modularity by improving code structure and design
- Reduced complexity to improve the maintainability of the source code
- Improves extensibility: Easier to add new features
- Refactoring involves improving internal structure of a software without altering its external behavior
- Goal of refactoring is to make the software more flexible and reusable
- It involves changing the design of the software after it has been coded to improve the design of a software. Refactoring is mainly used to improve badly designed software.
- Ensure software's external behavior remains the same after refactoring
- Refactoring is a valid activity only when the code is already in working state



Some of the Code Issues are:

•Lack of Reusable components : Potentially component may not be reused.

•Poor Coding Style : Coding standards/naming conventions has not been followed

•Lack of Modularity : Application components can't be reused as it is tightly coupled.

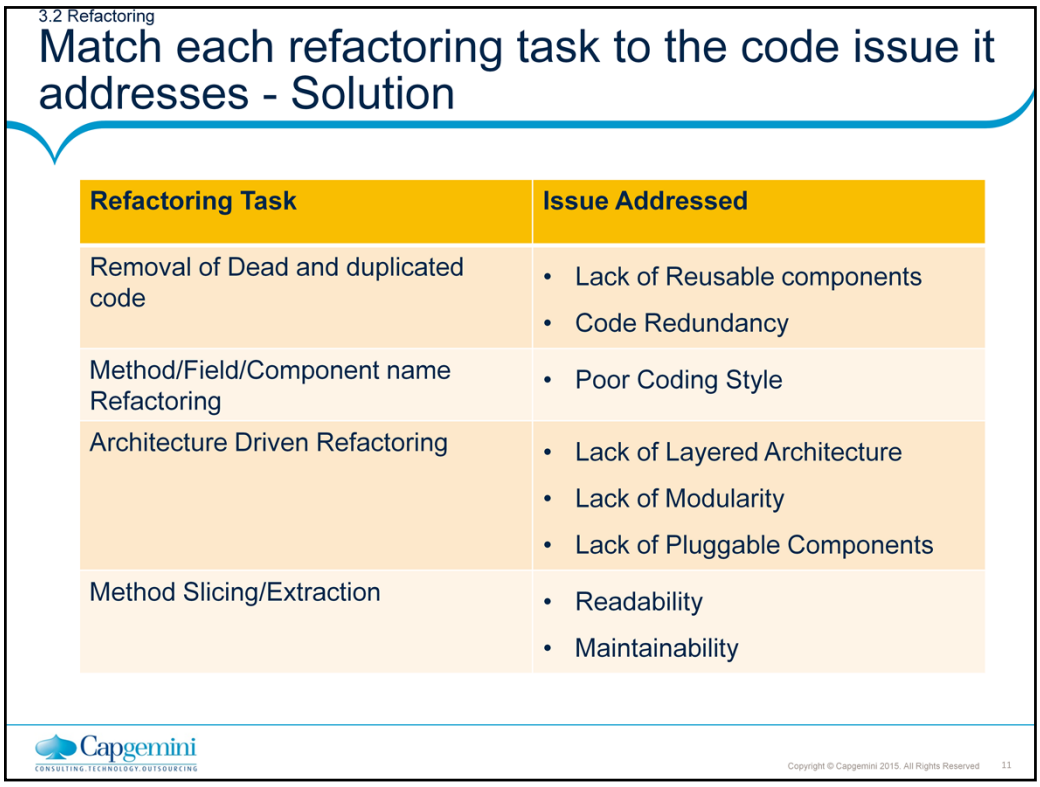
•Code Redundancy : Duplicate code and dead code exists.

•Lack of Layered Architecture : Any change in one layer causing changes in all other layers

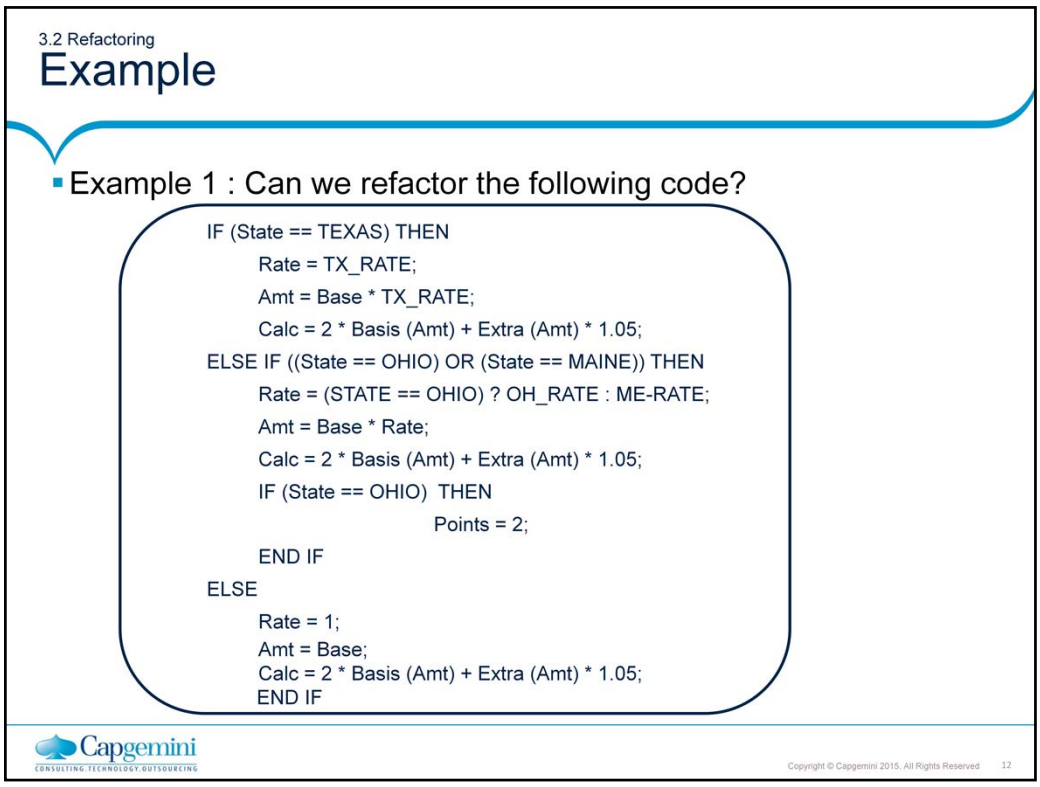
•Readability : Code is not readable due to lack of meaningful name and comments.

•Lack of Pluggable Components : Existing components are not easily replaceable due to the tight coupling code.

•Maintainability: Hard to maintain the code.



Each refactoring task is addressed to the code issue it addresses.

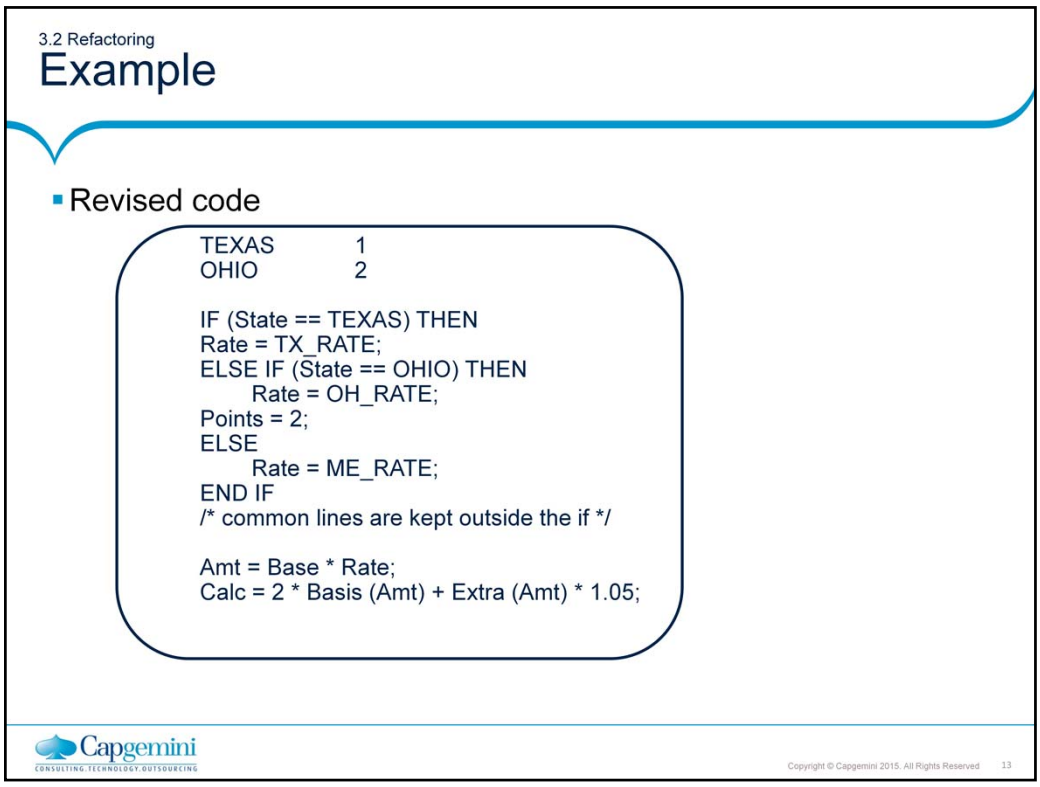


The above code is used to calculate tax based on the state. Tax rate varies for each state but similarity exists in the tax calculation.

Rate, Amt and Calc variables are assigned values in each if-else condition.

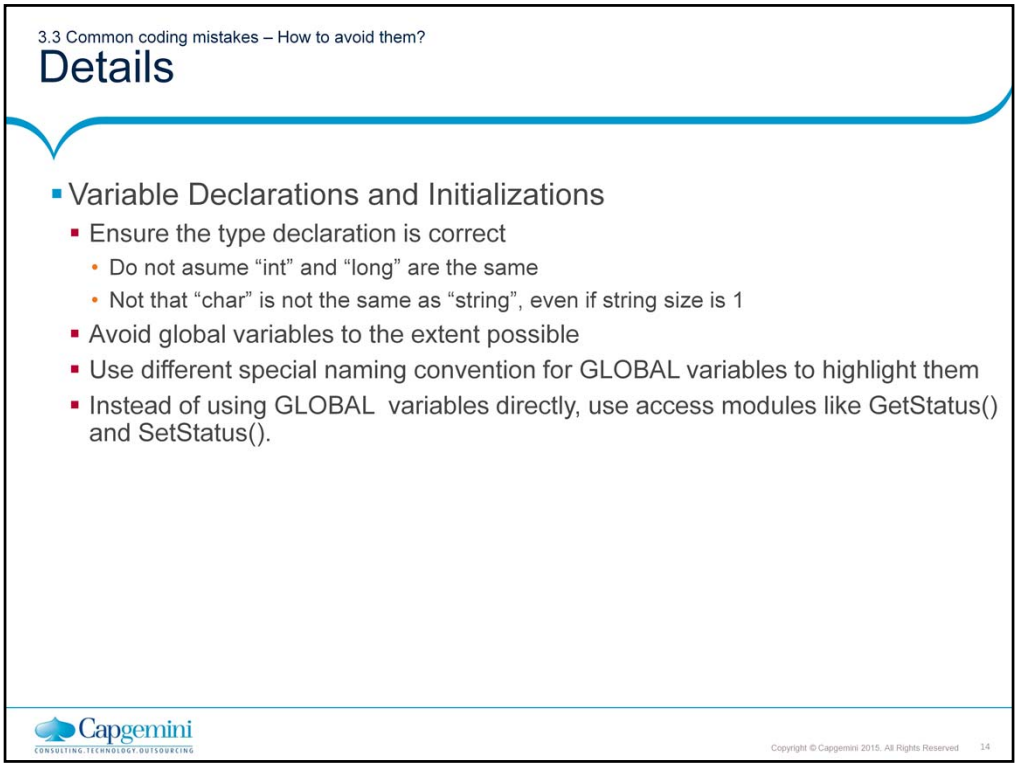
Use Refactoring for performing the below task:

- Avoid Duplicate code in each if else condition
- Common code should be kept outside if condition



Note: The code is more efficient and readable as compared to the previous code.

Are any further improvements possible?
If the program is used for three states, it is likely to be used for other states, as well, in the future.
Instead of using a nested IF ELSE statement, using a table or an array for tax rates of all states will make it easy to add more states later on.



Variable Declarations and Initializations:

Code Snippet - Example:

```
Boolean more-records = TRUE;
…
while ( more-records ) do …
```

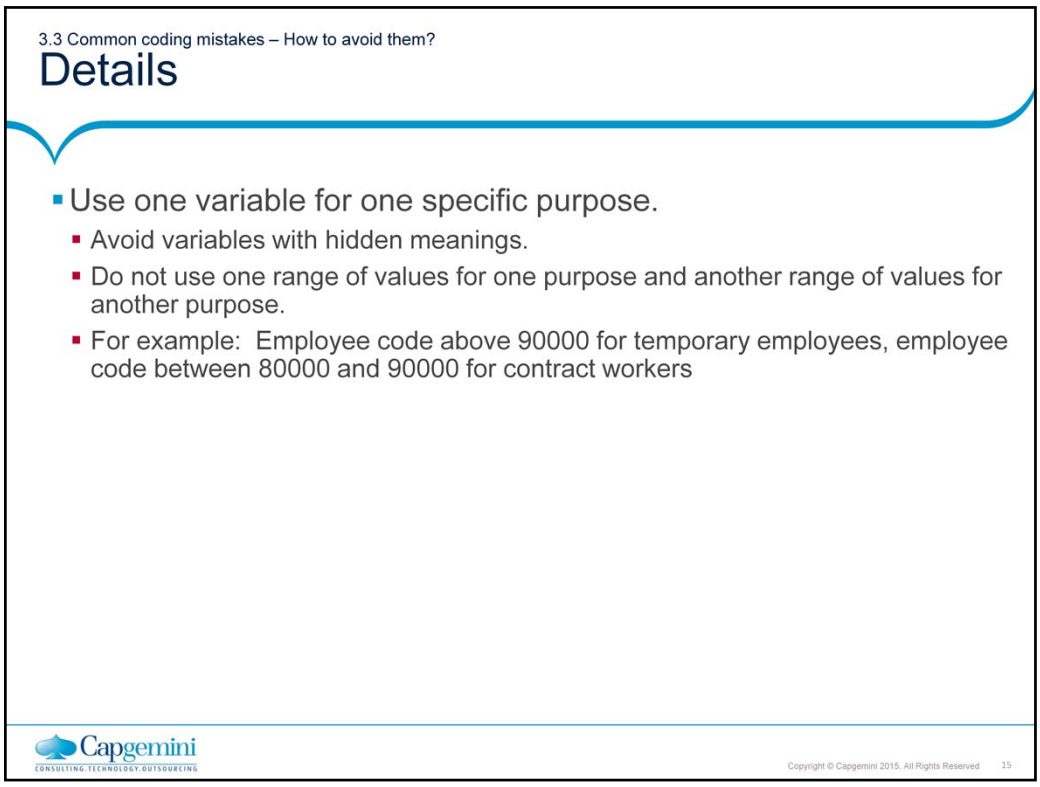
Better version of the above code snippet:

```
Boolean more-records;
…
more-records = TRUE;
while ( more-records ) do …
```

Avoid GLOBAL variables to the extent possible. : Note that it is difficult to debug, because a GLOBAL variable can be changed from any module. It may create problems with a recursive module.

Use different special naming convention for GLOBAL variables to highlight them. For example: MAX_USERS_G for global

Instead of using GLOBAL variables directly, use access modules like GetStatus() and SetStatus(). Note that this ensures that the variable is used only at once place inside those modules. This becomes easier to change or debug.

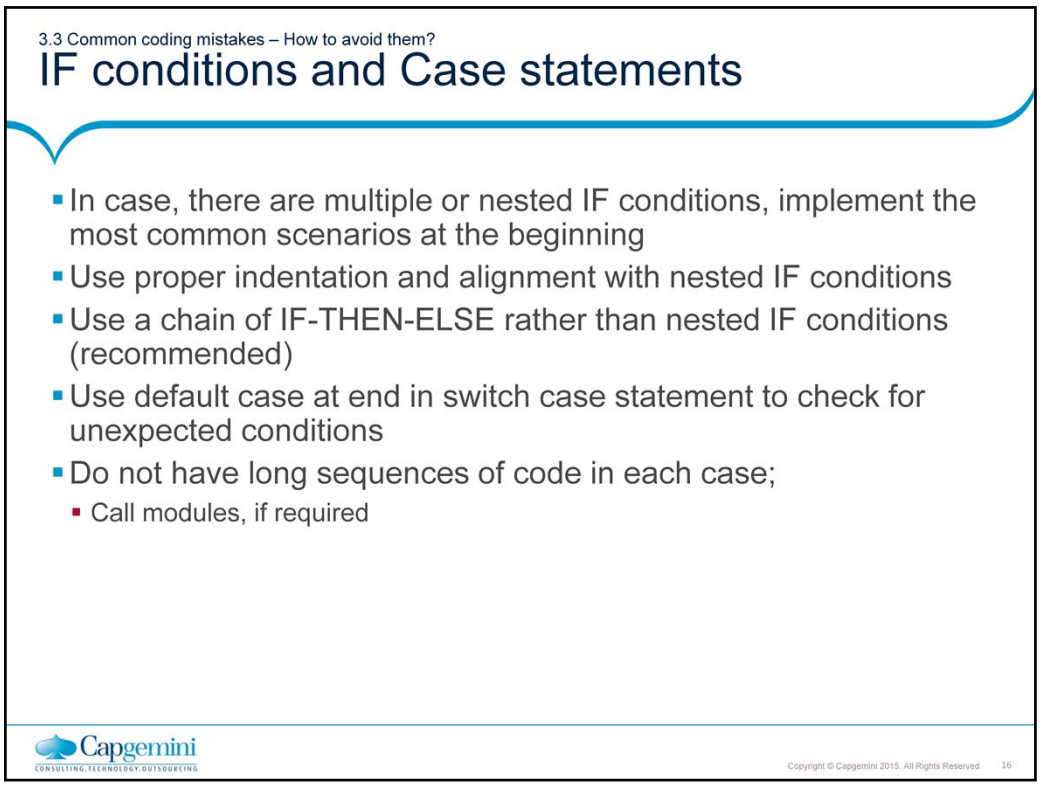


Use one variable for only one purpose.

For an example, if a variable is used for multiple different what can we provide? Consider a variable that is used to count the number of students, and count their grades. How would you name this variable: count, studentCount/gradeCount?

So create one variable for one purpose, otherwise data management related to the requirement will be complex.

- **Don't use variable with hidden meanings** because it might not be understandable by others.

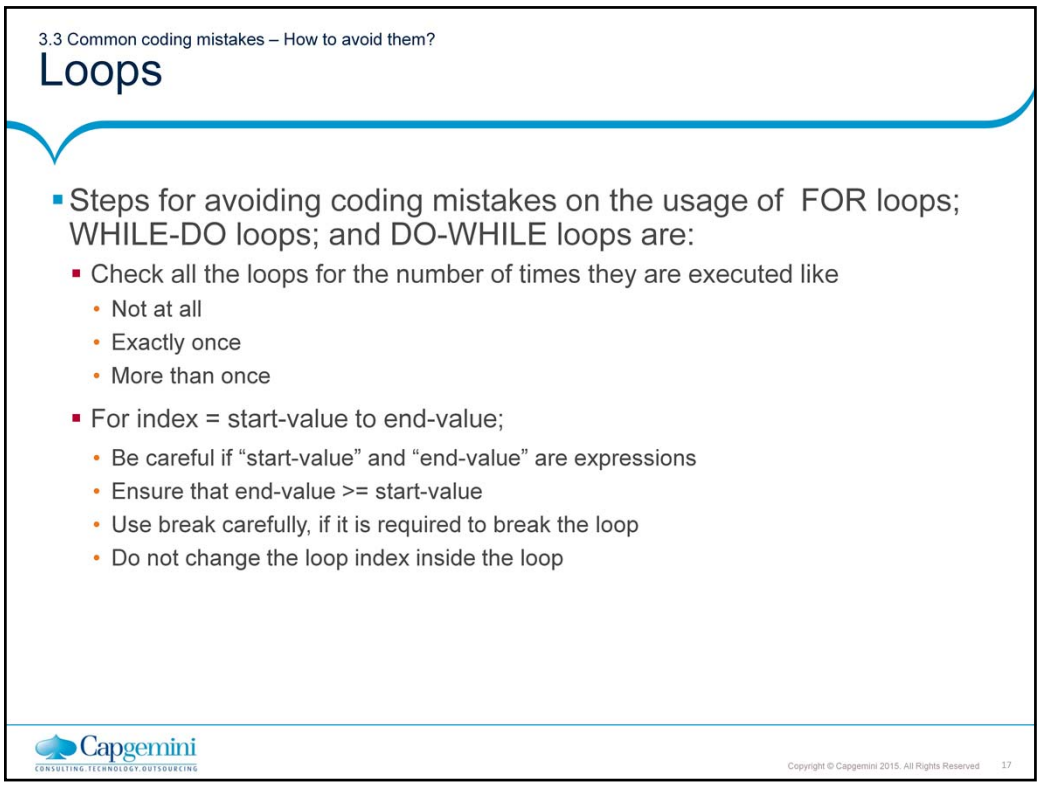


How to avoid Common Coding Mistakes while using IF and case statements:

- If there are multiple nested IF conditions, then find out the condition which has high priority and specify the same in first if condition.
- Adopt a standard indentation style for your code, and stick with it throughout the program so that program will be easily readable by other programmers.
- **For an Example:**

```
BEGIN
    DECLARE file AS FILE
    PROMPT "Enter the filename" AND STORE IN file
    IF (theFileExits) THEN
            determine the length of the file
             IF(FileLength>o) THEN
                readFile(file); // Invoke readFile module
             ELSE
                PRINT "File doesn't contain data"
              END IF
    ELSE
                PRINT "File doesn't exist"
    END IF
END
```

In the above code, indentation is followed, if-else is used and fileexists condition is checked first before finding length of a file for avoiding common mistakes.



Steps for avoiding Common Coding Mistakes while using Loops are:

- Check for the number of times the loop executed: For an example, execute the below loop thrice, by considering num values as 5, 4 and 2.

```
WHILE (num<5)
DO
 ….
END WHILE
```

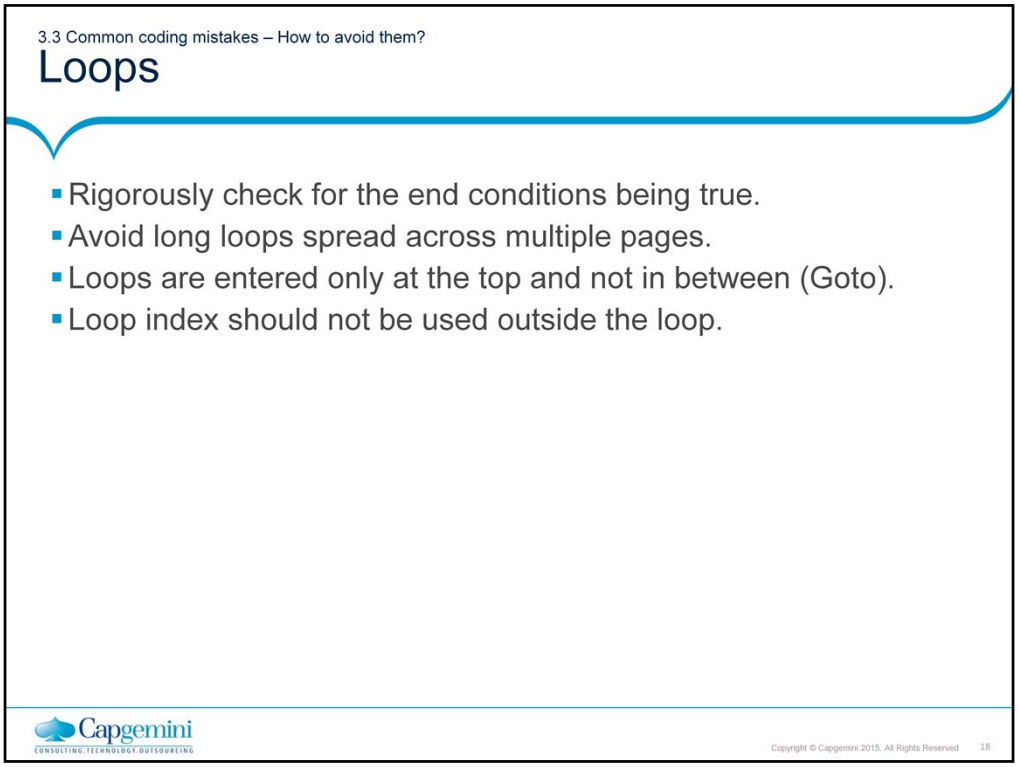
The above loop will be executed once for the value of 4, and the loop will not be executed at all for the num value of 5 and the same loop will be executed more than once if num value is 2.

- For loop:
  - Ensure that endvalue>=startvalue.

Better version	Not recommended to be used
FOR index= 0 to 10	FOR index=10 to 0
..	...
END LOOP	END LOOP

  - Do not change the loop index within the loop as shown in the below example. In the below example, loop execution will be stopped after the re-initialization of loop index inside the loop.

```
FOR index = 0 to 10
        index=13
END LOOP
```



Steps for avoiding Common Coding Mistakes while using Loops are:

Rigorously check for the end conditions being true : Rigorously avoid coding infinite loops.

Avoid long loops spread across multiple pages : The loop should be short enough to view all loop code at once.

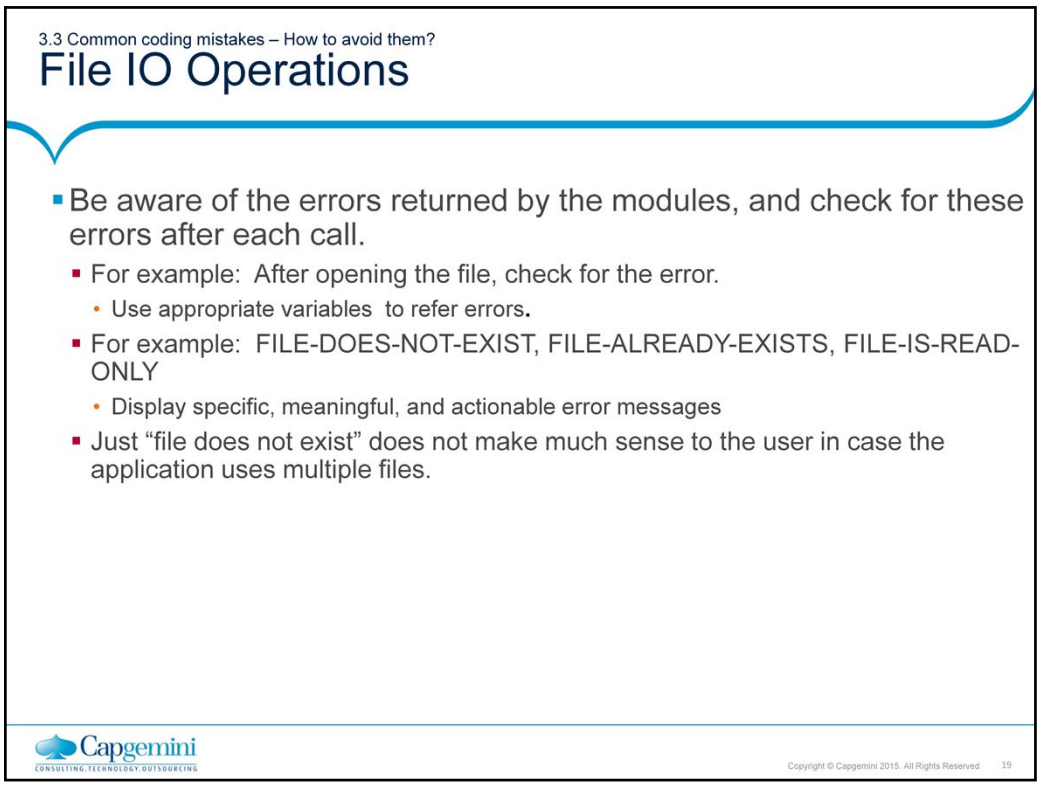
Loops are entered only at the top and not in between (Goto) : “goto” statement is always not recommended to be used inside loop as the loop execution always re start from the lower limit.

Loop index should not be used outside the loop. Consider the below code

```
FOR index=1 to 5
END LOOP
index=3;
```

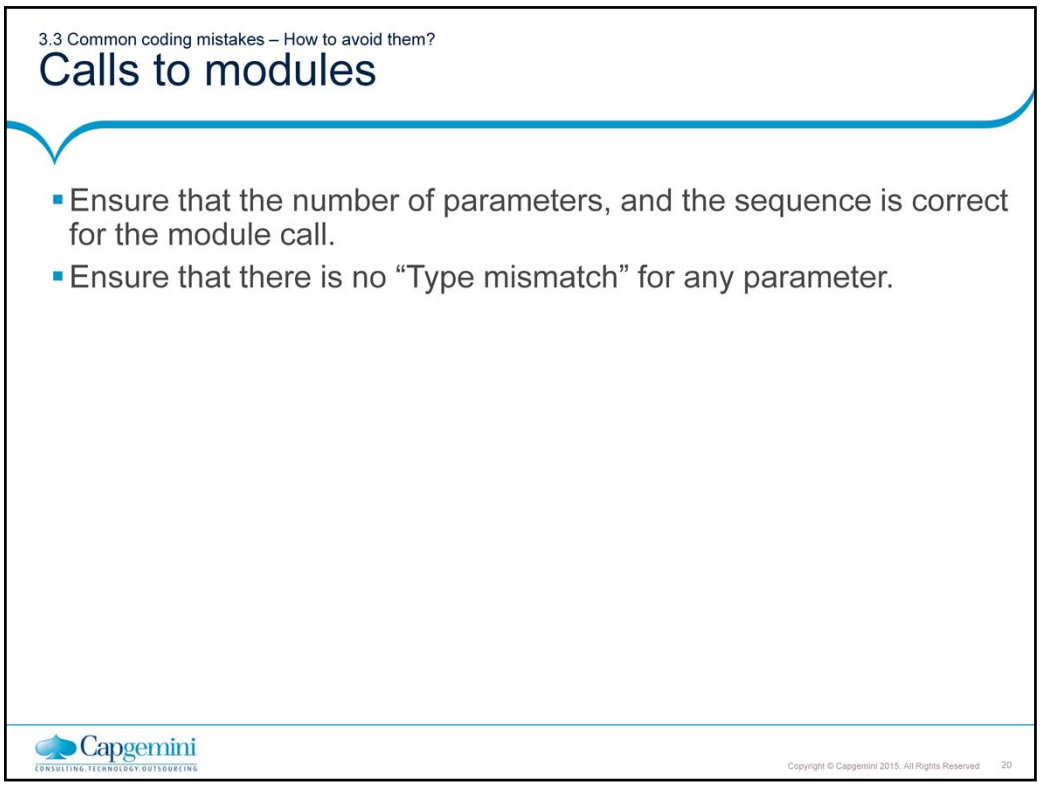
Don’t use index outside of the loop. Use another variable, if required as shown below:

```
FOR index=1 to 5
END LOOP
num=3;
```



Points to be considered for avoiding Common Coding Mistakes while performing file IO operations are:

- Be aware of the errors returned by the modules, and check for these errors after each call like
    - Check for the error, if file doesn’t exists
    - After opening the file, check for the error.
    - Check for the error, if file size is zero.
    - Check for the error, if file is not readable/writable.
- Use appropriate variables to refer errors. For example: FILE-DOES-NOT-EXIST, FILE-ALREADY-EXISTS, FILE-IS-READ-ONLY
- Display specific, meaningful, and actionable error messages. For an Example, Just “file does not exist” does not make much sense to the user in case the application uses multiple files. Instead use the error message as “Employee.txt file does not exist”, considering “Employee.txt” is a filename.



Consider the below signature of a module to calculate total price.

calculateTotal(Integer price, Integer quantity)

Refer the valid and invalid statements to invoke a module

calculateTotal(3,5); //Valid

calculateTotal(4,3,4); //Invalid

calculateTotal(‘Test’,3); //Invalid

Summary

- In this lesson, you have learnt about:
  - Record is a composite data type used to store data of different datatypes
  - Record is a composite data Usage of files
  - Refactoring is the process of clarifying and simplifying the design of existing code, without changing its behavior.
  - Use proper variable types, and initialize wherever necessary.
  - Avoid GLOBAL and STATIC variables to the best extent possible.
  - In case of multiple or nested IF conditions, implement the most common scenarios at the beginning
  - In case of loops, rigorously check whether the end conditions are true.
  - In case of files, be aware of the errors returned by the functions, and check for these errors after each call.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Review Question

Question 1: Mixing different types of calculation is a good practice.

A. True
B. False

Question 2: Which of these are characteristic benefits of refactoring

A. It enables you to add functionality to your code

B. Its main function is to optimize performance

C. It makes code easier to understand and modify

D. It makes code more flexible and reusable

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Question

Question 3: Data item that make up structure can be of the different types.

A. True
B. False

Question 4: Which type of file format is used to store one character per byte?

A. Binary File
B. Text File

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23