Core Java Fundamentals

Capgemini

Table of Content

What you will learn

Key skills you will gain upon completion of this program include:

- Understanding & working with basic construct, OOPs & Data Structure.
- Handling Errors and Exception and working with in-built classes.
- Working with collections & choosing right collection in a scenario.
- Working with files, understanding and using important concepts like Serialization.
- Working with Java 8 features like Lambdas, Stream & Date API
- Understanding and implementing Layered architecture with JDBC API for different databases.
- Understanding best practices and standards.
- Logging and unit testing (Junit Testing Framework) and Junit must have 70% coverage.
- Build a banking application, Working with Annotation and Custom annotation in Java.
- Mockito test framework used to mock dependencies in unit tests.
- Learn to analyze the code, check coding standard and work with Java Code coverage tool available free in market.

Software requirement

- JDK 8.x
- Eclipse(https://www.eclipse.org/downloads/)
- Junit plugins for Eclipse
- Mockito Framework
- MySQL /Oracle for database communication

Pre-requisites

The participants need not have any prior exposure to Java programming language.

Prior familiarity with some programming language (Such as C or C++) would be useful.

Knowledge and understanding of SQL is mandatory.

M001:Language Fundamentals

Day 1: Language Fundamentals

In this module , you will learn about java architecture, advantages of Java, develop the code with various datatypes, conditions, loops and arrays.

Day Wise Schedule

M001: Language Fundamentals
Day 1
Getting Started

- Introduction to Java
- Writing, compiling and running a program
- Platform Independency in Java
- Integrated Development Environment
- Some important terms in Java

Basic Language Constructs

- Naming conventions in Java
- Variables and data types
- Operators (arithmetic, assignment, relational, logical and bitwise)
- Promotion and demotion rules for operators
- Looping (while, do…while, for loops)
- Conditional statements (if…else…, switch case)
- break and continue statements
- Reference Variables

Day Wise Schedule

Implementing Data Structure

- Arrays
- Linked List
- Stack
- Heap
- Tree

Useful References

Java Introductions
https://www.vogella.com/tutorials/JavaIntroduction/article.html

Basic Assignment

double click on image

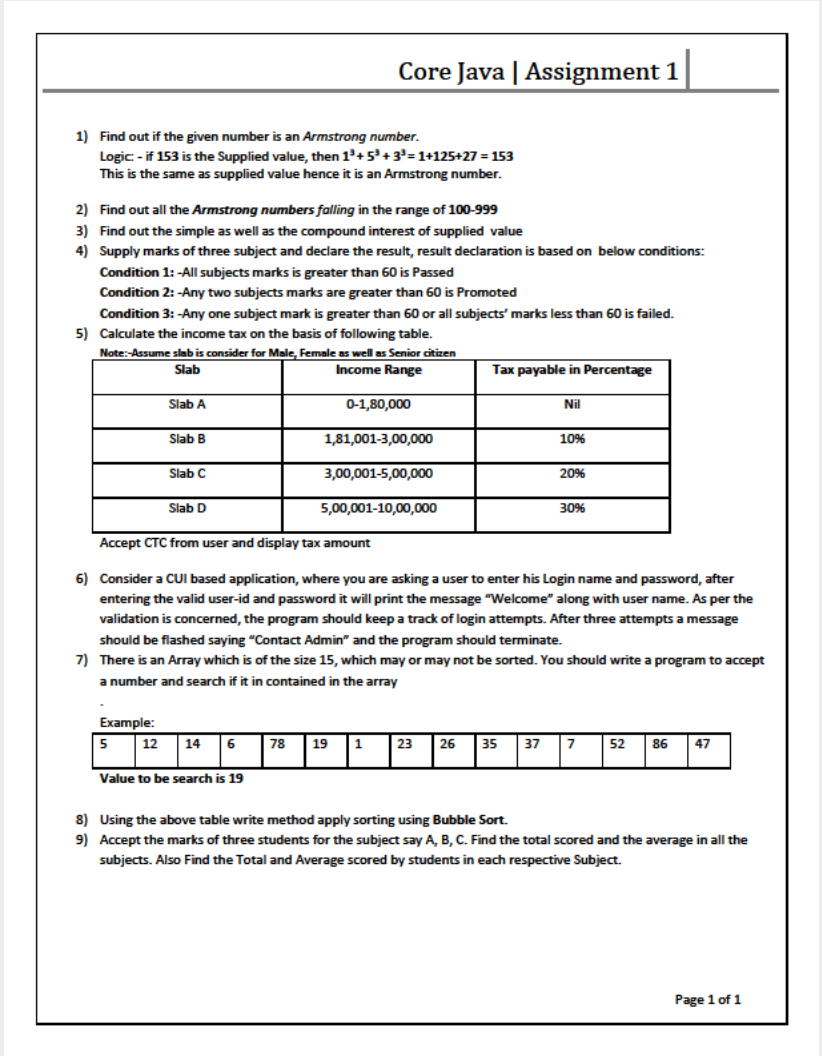
1. Write a program to display Hello World on console.
2. Write a program to print all command line arguments.
3. Pass integer numbers through command line & display them in sorted fashion.
4. Write the class Date having attributes like day, month & year. Add default & parameterized constructors. Add getters & setters. Add method to print the date. Add method to swap two dates.
5. Write a class ComplexNumber having attributes real & imaginary. Add functions like add, subtract, multiply & swap.
6. Write a class Account & add methods like deposit, withdraw, print etc.
7. Supply marks of three subject and declare the result, result declaration is based on below conditions:
    a. Condition 1: -All subjects marks is greater than 60 is Passed
    b. Condition 2: -Any two subjects marks are greater than 60 is Promoted
    c. Condition 3: -Any one subject mark is greater than 60 or all subjects' marks less than 60 is failed.
8. Write java classes to build doubly linked list. Add functionalities like add new node, insert node, delete node, count nodes & print linked list.
9. Write a program to implement a Stack using arrays using the following-

```
class StackedArray {
        int ary[];
        push(--) { }
        pop() {--) {}
}
```

10. Write a program to implement a Queue using arrays using the following-

Assignment

double click on image



Core Java | Assignment 1

1) Find out if the given number is an *Armstrong number*.
   Logic: - if **153** is the Supplied value, then $1^3 + 5^3 + 3^3 = 1+125+27 = 153$
   This is the same as supplied value hence it is an Armstrong number.

2) Find out all the ***Armstrong numbers falling*** in the range of **100-999**
3) Find out the simple as well as the compound interest of supplied value
4) Supply marks of three subject and declare the result, result declaration is based on below conditions:
   Condition 1: -All subjects marks is greater than 60 is Passed
   Condition 2: -Any two subjects marks are greater than 60 is Promoted
   Condition 3: -Any one subject mark is greater than 60 or all subjects' marks less than 60 is failed.
5) Calculate the income tax on the basis of following table.
   Note:-Assume slab is consider for Male, Female as well as Senior citizen

Slab	Income Range	Tax payable in Percentage
Slab A	0-1,80,000	Nil
Slab B	1,81,001-3,00,000	10%
Slab C	3,00,001-5,00,000	20%
Slab D	5,00,001-10,00,000	30%

   Accept CTC from user and display tax amount

6) Consider a CUI based application, where you are asking a user to enter his Login name and password, after entering the valid user-id and password it will print the message "Welcome" along with user name. As per the validation is concerned, the program should keep a track of login attempts. After three attempts a message should be flashed saying "Contact Admin" and the program should terminate.
7) There is an Array which is of the size 15, which may or may not be sorted. You should write a program to accept a number and search if it in contained in the array

   .
   Example:

5	12	14	6	78	19	1	23	26	35	37	7	52	86	47

   Value to be search is 19

8) Using the above table write method apply sorting using **Bubble Sort**.
9) Accept the marks of three students for the subject say A, B, C. Find the total scored and the average in all the subjects. Also Find the Total and Average scored by students in each respective Subject.

M002:Object Oriented Programming in Java

Day 2: Object Oriented Programming in Java

In this module , you will learn about Object oriented programming through Classes, Objects and various concepts like Abstract , Final, relationship, interface , inheritance and Polymorphism etc.

Day Wise Schedule

M002: Object Oriented Programming in Java
Day 2:
Classes and objects

- Access control
- Constructor and Init block
- Overloading
- Static methods and fields
- Garbage collection -finalize() method
- The toString method

Association Relationship

- Association
- Aggregation
- Composition
- Relationship exercising using setters/constructors

Day Wise Schedule

Extending Classes
- Inheritance
- Protected keyword
- Constructors in extended classes
- Overriding methods
- Polymorphism
- Making Methods and Classes Final

Abstract Classes
- Abstract classes and methods
- Extending abstract class
- Abstract class and Polymorphism

Useful References

OOPS Concepts

https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

https://www.guru99.com/java-oops-concept.html

https://www.javatpoint.com/java-oops-concepts

Assignment

double click on image

1. Write a singletone class. Confirm that singletone class cannot be inherited.
2. Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes of the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use polymorphism i.e. method overriding.
3. Write a program to consider saving & current account in the bank. Saving account holder has 'Fixed Deposits' whereas Current account holder has cash credit. Apply polymorphism to find out total cash in the bank.
4. Test the following principles of an abstract class:
   - If any class has any of its method abstract then you must declare entire class abstract.
   - Abstract class cannot be instantiated.
   - When we extend an abstract class, we must either override all the abstract methods in sub class or declare subclass as abstract.
   - Abstract class cannot be private.
   - Abstract class cannot be final.
   - You can declare a class abstract without having any abstract method.
5. Write the classes Line, Rectangle, Cube etc. & make the Shape as their base class. Add an abstract draw() method in the class Shape & draw all shapes.
6. Write an abstract class 'Persistence' along with two sub classes 'FilePersistence' & 'DatabasePersistence'. The base class with have an abstract method persist() which will be overridden by its sub classes. Write a client who gets the Persistence object at runtime & invokes persist() method on it without knowing whether data is being saved in File or in Database.

M003:WorkBench Tools

Day 3:WorkBench Tools

In this module , you will learn how to integrate Checkstyle, PMD, Eclemma , SonarQube and code coverage tools in your java project.

Day Wise Schedule

M003: WorkBench tools

Day 3:

- Checkstyle Eclipse plug-in
  - Introduction to Checkstyle
  - Working with Maven Plugin
  - Exercise: Use the Checkstyle Maven plugin
- PMD Eclipse plug-in
  - Introduction to PMD
  - How to use PMD
  - Configure PMD plugin with eclipse
  - Generate PMD reports with eclipse
- EclEmma code coverage eclipse plug-in
  - Introduction to EclEmma
  - how to configure EclEmma in eclipse
  - eclipse code coverage JUnit
  - eclipse code coverage plugin
- FindBugs in Eclipse
  - What is FindBugs
  - Installation of the FindBugs tooling for Eclipse
  - Using FindBugs in Eclipse

Day Wise Schedule

- Code Analysis with SonarQube
  - Introduction to SonarQube
  - Analysing Source Code
  - Analysis Result
  - SonarQube Quality Gate
  - Integrating SonarQube into a CI

Useful References

Using the Checkstyle Eclipse plug-in – Tutorial
https://www.vogella.com/tutorials/Checkstyle/article.html

PMD Tutorial
https://www.javatips.net/blog/pmd-in-eclipse-tutorial

EclEmma code coverage eclipse plug-in
https://crunchify.com/what-is-the-best-code-coverage-plugin-you-should-use-in-eclipse-ide/

FindBugs in Eclipse – Tutorial
https://www.vogella.com/tutorials/Findbugs/article.html

Code Analysis with SonarQube
https://www.baeldung.com/sonar-qube

Assignment

Integrate PMD, Checkstyle and Ecl emma in to your current project

M004:Unit Testing and Logging

Day 4:Unit Testing and Logging

In this module , you will learn how to write and run unit test with Junit Testing Framework, learn variety of Junit ASSERT methods, parameterized testing, learn about the JUnit best practices and test the performance and Exceptions in unit tests.

Apply Logging API for recording application information effectively, Master logging levels and handlers.

Day Wise Schedule

M004: Unit Testing and Logging

- Day 4 :
  - Working with JUnit
    - The purpose of software tests
    - Testing terminology
    - Using JUnit
    - Using JUnit 4
    - Eclipse support for JUnit 4
    - Installation of JUnit
    - Setting Eclipse up for using JUnits static imports
    - Exercise: Using JUnit
    - Mocking
    - Overview of JUnit 5
    - Comparison of annotations between JUnit 4 and 5
  - Logging using Log4j
    - Log4j Configuration
    - Java Log4j usage
    - Log4j Java Web Application

Day Wise Schedule

- Logging using SLF4J
  - Logging Example
  - Java.Util.Logging
  - Logback
  - Performance and Considerations
  - Factory Methods
  - Logger, Appender and Levels
  - Mapped Diagnostic Context
  - Parameterized Logging
  - Implementing SLF4J

Useful References

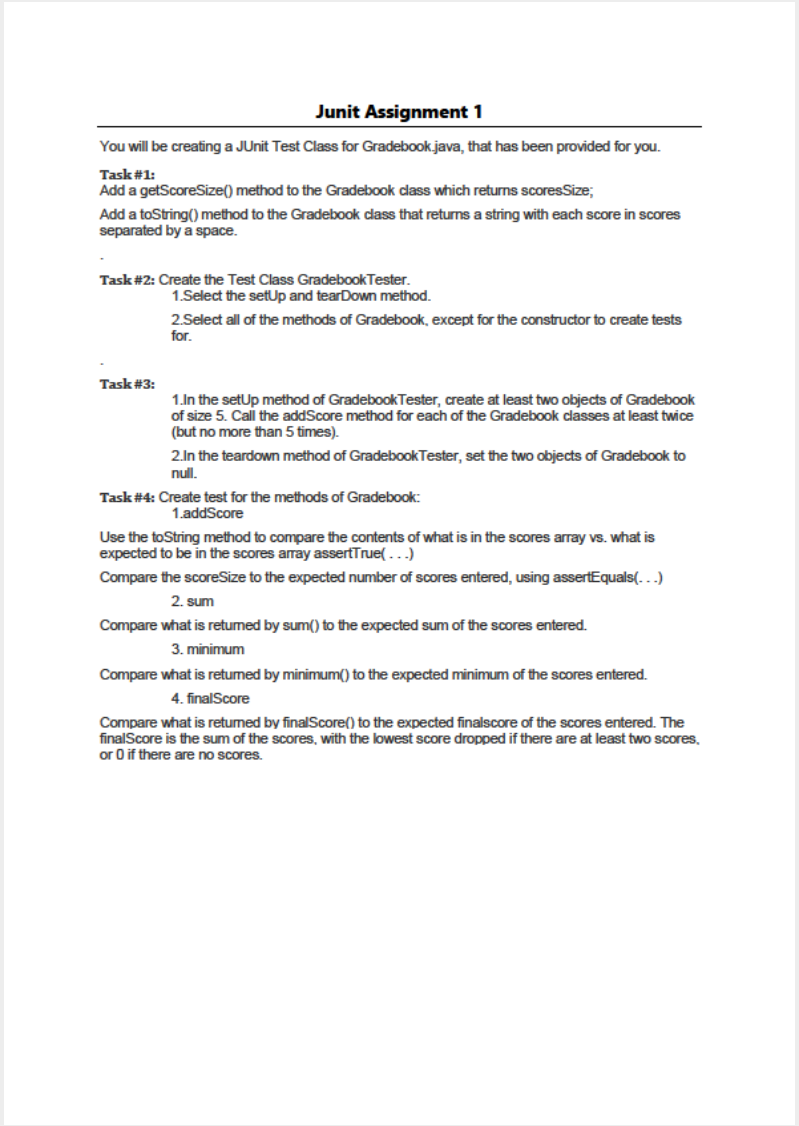
Junit Tutorial
https://www.vogella.com/tutorials/JUnit/article.html
https://www.youtube.com/watch?v=o5k9NOR9lrI

Logging using Log4j
https://www.journaldev.com/10689/log4j-tutorial
https://howtodoinjava.com/log4j/

Logging using SLF4J
https://examples.javacodegeeks.com/enterprise-java/slf4j/slf4j-tutorial-beginners/

Advanced Junit Assignment

double click on image



Junit Assignment 1

You will be creating a JUnit Test Class for Gradebook.java, that has been provided for you.

Task #1:
Add a getScoreSize() method to the Gradebook class which returns scoresSize;

Add a toString() method to the Gradebook class that returns a string with each score in scores separated by a space.

.

Task #2: Create the Test Class GradebookTester.

1.Select the setUp and tearDown method.

2.Select all of the methods of Gradebook, except for the constructor to create tests for.

.

Task #3:

1.In the setUp method of GradebookTester, create at least two objects of Gradebook of size 5. Call the addScore method for each of the Gradebook classes at least twice (but no more than 5 times).

2.In the teardown method of GradebookTester, set the two objects of Gradebook to null.

Task #4: Create test for the methods of Gradebook:

1.addScore

Use the toString method to compare the contents of what is in the scores array vs. what is expected to be in the scores array assertTrue(. . .)

Compare the scoreSize to the expected number of scores entered, using assertEquals(. . .)

2. sum

Compare what is returned by sum() to the expected sum of the scores entered.

3. minimum

Compare what is returned by minimum() to the expected minimum of the scores entered.

4. finalScore

Compare what is returned by finalScore() to the expected finalscore of the scores entered. The finalScore is the sum of the scores, with the lowest score dropped if there are at least two scores, or 0 if there are no scores.

M005:Exception Handling and Built in Classes

Day 5: Exception Handling and Built in Classes

In this module, you will learn basics to advanced exception handling in Java, Mostly occurred java exception and errors, Handle errors in java with try-catch technique and write custom exception classes for handling errors.

Able to work with String , String Buffer & String Builder classes.

Day Wise Schedule

- M005: Exception Handling and Build in Classes
- Day 5:
  - Exception Handling
    - Checked exceptions
    - Unchecked exceptions
    - The “try-throw-catch” structure
    - The “finally” clause
    - Custom Exception
    - Exception chaining
    - New Features of Java 7
    - Try with Resources
    - AutoCloseable
    - Catch Block Handling Multiple Exceptions
  - Some Useful In-Built Classes
    - The Object class
    - The String class
    - The String Buffer class
    - The StringBuilder class

Useful References

Exception handling
https://www.journaldev.com/1696/exception-handling-in-java

String classes
https://www.journaldev.com/538/string-vs-stringbuffer-vs-stringbuilder

Assignment

double click on image

1. **EXCEPTION HANDLING:** Write a user defined exception called 'InsufficientBalanceException'. Use this exception in withdraw() method of class Account. Test it by making the exception checked & then unchecked.
2. **EXCEPTION HANDLING:** Write a user defined auto closable class & test its close method invocation by using the try with resources.
3. **EXCEPTION HANDLING**: Test different forms of try, catch & finally:
    a. Try with multiple catch
    b. Nested try/ catch blocks
    c. Try, catch & finally
    d. Try with finally

M006:Wrapper Classes and Serialization

Day 6: Wrapper Classes and Serialization

In this module, you will learn to work with wrapper classes and serialization techniques.

Day Wise Schedule

- M006: Wrapper Classes and serialization
- Day 6:
  - Wrapper classes and Auto-Boxing
    - Enumeration
    - Wrapper classes
    - Auto Boxing and Un-Boxing
  - Java Serialization
    - Types of Input and Output Streams
    - Byte-based stream
    - The Challenge of Object Serialization
    - Serialization API
    - Serializable Interface
    - ObjectInputStream and ObjectOutputStream
    - The Serialization Engine
    - Transient Fields
    - Serialization in Inheritance

Useful References

Wrapper classes
https://www.geeksforgeeks.org/wrapper-classes-java/

Serialization
https://www.geeksforgeeks.org/serialization-in-java/

Assignment

1. Create a list and fill it with 20 random numbers from this range: `(10, 15, 20, 25, 30, 35, 40, 45, 50)`
   Seed the constructor of class Random with the value 19 (this way your output will match the output provided)
2. Print the list
3. Sort the numbers in the list. (the list is now modified)
   Then serialize the list to a file called Number`List.ser`
4. Deserialize the file into a new list called `numbers2`. Print it.
5. Reposition the elements of `numbers2` so that they are in random order (`numbers2` is now modified).
   Print it again

Make the output look like the output provided

Output:

```
numbers : [35, 20, 35, 25, 15, 20, 50, 10, 50, 30, 30, 25, 10, 30, 30, 10, 45, 25, 45, 20]
numbers2: [10, 10, 10, 15, 20, 20, 20, 25, 25, 25, 30, 30, 30, 30, 35, 35, 45, 45, 50, 50]
numbers2: [20, 50, 45, 10, 30, 10, 25, 15, 50, 30, 35, 10, 20, 30, 25, 45, 35, 30, 20, 25]
```

M007 & 8: Java Generics and Collection Framework

Day 7: Java Generics and Collection Framework

In this module, you will learn to understand the basics of generics understand the basic data structure ,Understand how to use different classes/interfaces of Java Collections Framework and implement generic algorithms.

Day Wise Schedule

M007: Java Generics and Collection Framework

- Day 7:
  - Java Generics
    - Generics in Java
    - Java Generic Class
    - Java Generic Interface
    - Java Generic Type
    - Java Generic Method
    - Java Generics Bounded Type Parameters
    - Java Generics and Inheritance
    - Java Generic Classes and Subtyping
    - Java Generics Wildcards
      - Java Generics Upper Bounded Wildcard
      - Java Generics Unbounded Wildcard
      - Java Generics Lower bounded Wildcard
    - Subtyping using Generics Wildcard
    - Java Generics Type Erasure

Day Wise Schedule

- Collection Framework
  - What is Java Collections Framework?
  - Benefits of Java Collections Framework
- Java Collections Interfaces
  - Collection Interface
  - Iterator Interface
  - Set Interface
  - List Interface

Day 8: Java Generics and Collection Framework

In this module, you will learn to understand the basics of generics understand the basic data structure ,Understand how to use different classes/interfaces of Java Collections Framework and implement generic algorithms.

Day Wise Schedule

M008: Java Generics and Collection Framework

Day 8:

- Java Collections Interfaces
  - Queue Interface
  - Dequeue Interface
  - Map Interface
  - ListIterator Interface
  - SortedSet Interface
  - SortedMap Interface
- Java Collections Classes
  - HashSet Class
  - TreeSet Class
  - ArrayList Class
  - LinkedList Class
  - HashMap Class
  - TreeMap Class

Day Wise Schedule

- Collections API Algorithms
  - Sorting
  - Shuffling
  - Searching
  - Composition
  - Min and Max values

Useful References

Generics in Java
https://www.journaldev.com/1663/java-generics-example-method-class-interface

Collections in Java
https://www.journaldev.com/1260/collections-in-java-tutorial

Assignment

double click on image

Collection Framework

In this assignment, you are required to write a menu-driven Java program that allows the user to add patients to a priority queue, display the next patient (and remove him/her from the queue), show a list of all patients currently waiting for treatment, and exit the program. The program should simulate the scheduling of patients in a clinic. Use the attached `Patient` class provided along with this assignment.

Your program should schedule patients in the queue according to the emergency of their cases from 1 to 5 (the higher the value, the higher the priority). If two patients have the same emergency value, use their order of arrival to set their priority (the lower the order, the higher the priority). It is up to you to have the `Patient` class implement either `Comparable` or `Comparator`.

Create a class `PatientManager` that has an attribute named `waitingList` of the type `PriorityQueue<Patient>` and a public method, `start()`. When `start` is called, it should display the following menu of choices to the user, and then ask the user to enter a choice from 1 to 4:

```
-------------------------
(1) New Patient.
(2) Next Patient.
(3) Waiting List.
(4) Exit.
-------------------------
```

Here is a description of each choice:

(1) Ask the user for the patient's name and the emergency from 1 to 5 (1 = low, and 5 = life-and-death). Your program should create an instance of `Patient` using the entered data and add it to the priority queue. Note that your program should not ask the user for the value of the patient's order of arrival. Instead, it should use a counter that is automatically incremented whenever a patient is added to the queue.
(2) Display the name of the next patient in the priority queue and remove him/her from the queue.
(3) Display the full list of all patients that are still in the queue.
(4) End the program.

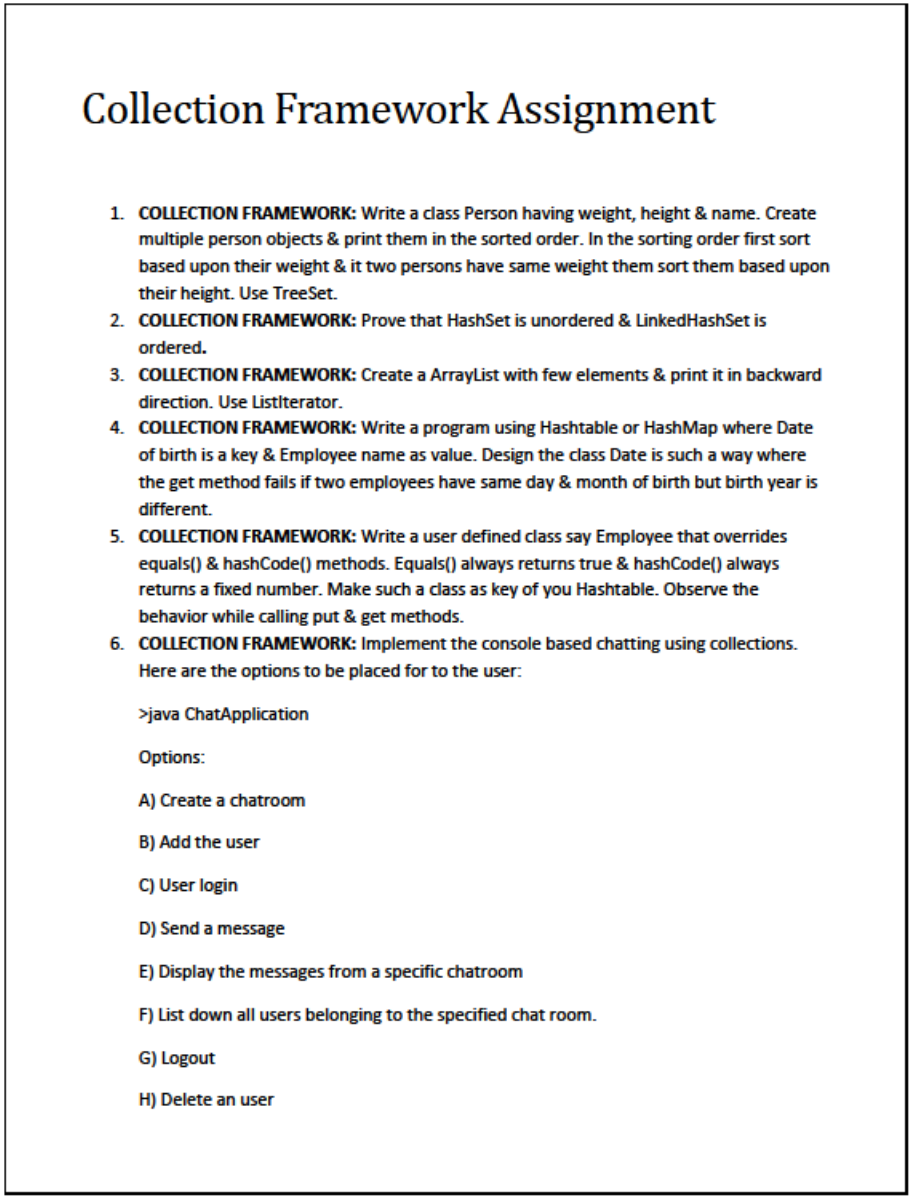
Make sure your `PatientManager` class is robust. It should not crash when a user enters invalid value. Instead, it should display an error message followed by an action depending on the type of error (see the sample run below).

Test your program by instantiating your `PatientManager` class in a `main` method and calling the `start` method.

Note: Add more helper methods and attributes as needed to the `PatientManager` class.

Assignment

double click on image



Collection Framework Assignment

1. **COLLECTION FRAMEWORK:** Write a class Person having weight, height & name. Create multiple person objects & print them in the sorted order. In the sorting order first sort based upon their weight & it two persons have same weight them sort them based upon their height. Use TreeSet.
2. **COLLECTION FRAMEWORK:** Prove that HashSet is unordered & LinkedHashSet is ordered.
3. **COLLECTION FRAMEWORK:** Create a ArrayList with few elements & print it in backward direction. Use ListIterator.
4. **COLLECTION FRAMEWORK:** Write a program using Hashtable or HashMap where Date of birth is a key & Employee name as value. Design the class Date is such a way where the get method fails if two employees have same day & month of birth but birth year is different.
5. **COLLECTION FRAMEWORK:** Write a user defined class say Employee that overrides equals() & hashCode() methods. Equals() always returns true & hashCode() always returns a fixed number. Make such a class as key of you Hashtable. Observe the behavior while calling put & get methods.
6. **COLLECTION FRAMEWORK:** Implement the console based chatting using collections. Here are the options to be placed for to the user:

   >java ChatApplication

   Options:

   A) Create a chatroom

   B) Add the user

   C) User login

   D) Send a message

   E) Display the messages from a specific chatroom

   F) List down all users belonging to the specified chat room.

   G) Logout

   H) Delete an user

M009: Java 8 Fundamentals

Day 8: Java 8 Fundamentals

In this module, you will learn to understand the new features introduced in Java 8, write lambda expressions and create custom lambda functions. Apply method references to write compact lambda expression and apply functional programming basics to java code.

Day Wise Schedule

- M009:Java 8 Fundamentals
- Day 9:
  - Java 8 Interface
    - Java Interface Default Method
    - Java Interface Static Method
  - Functional Interfaces
    - Introduction to Functional interface
    - Java 8 Functional Interface
    - Lambda Expression
    - Why do we need Lambda Expression
    - Lambda Expression Examples
    - Method and Constructor References

Useful References

Java 8 Interface
https://www.journaldev.com/2752/java-8-interface-changes-static-method-default-method

Functional Interface
https://www.journaldev.com/2763/java-8-functional-interfaces

Assignment

double click on image

Lambda expression assignments

1. Write an application to perform basic arithmetic operations like add, subtract, multiply & divide. You need to define a functional interface first.
2. Write an application using lambda expressions to print Orders having 2 criteria implemented: 1) order price more than 10000 2) order status is ACCEPTED or COMPLETED.
3. Use the functional interfaces Supplier, Consumer, Predicate & Function to invoke built-in methods from Java API.
4. Remove the words that have odd lengths from the list. HINT: Use one of the new methods from JDK 8. Use removeIf() method from Collection interface.
5. Create a string that consists of the first letter of each word in the list of Strings provided. HINT: Use Consumer interface & a StringBuilder to construct the result.
6. Replace every word in the list with its upper case equivalent. Use replaceAll() method & UnaryOperator interface.
7. Convert every key-value pair of the map into a string and append them all into a single string, in iteration order. HINT: Use Map.entrySet() method & a StringBuilder to construct the result String.
8. Create a new thread that prints the numbers from the list. Use class Thread & interface Consumer.

M010: Java 8 Streams

Day 10: Java 8 Stream

In this module , you will understand the benefits streams have to offer , use streams to perform transformations using stream().map(…), know how to use Java Streams API.

Day Wise Schedule

- M0010: Java 8 Streams
- Day 10:
  - Java 8 Stream
    - Collections and Java Stream
    - Functional Interfaces in Java 8 Stream
      - Function and BiFunction
      - Predicate and BiPredicate
      - Consumer and BiConsumer
      - Supplier
    - java.util.Optional
    - java.util.Spliterator
    - Java Stream Intermediate and Terminal Operations
    - Java Stream Short Circuiting Operations
    - Java Stream Examples
      - Creating Java Streams
      - Converting Java Stream to Collection or Array
      - Java Stream Intermediate Operations
      - Java Stream Terminal Operations
    - Java 8 Stream API Limitations

Useful References

Java Stream
https://www.journaldev.com/2774/java-8-stream

Assignment

double click on image

Streams assignments

Setup:

Create the following classes:

class Fruit { String name; int calories; int price; String color; }

class News { int newsId; String postedByUser; String commentByUser; String comment; }

class Trader { String name; String city; }

class Transaction { Trader trader; int year; int value; }

1. Display the fruit names of low calories fruits i.e. calories < 100 sorted in descending order of calories.
2. Display color wise list of fruit names.
3. Display only RED color fruits sorted as per their price in ascending order.
4. Find out the newsId which has received maximum comments.
5. Find out how many times the word 'budget' arrived in user comments all news.
6. Find out which user has posted maximum comments.
7. Display commentByUser wise number of comments.
8. Find all transactions in the year 2011 and sort them by value (small to high).
9. What are all the unique cities where the traders work?
10. Find all traders from Pune and sort them by name.
11. Return a string of all traders' names sorted alphabetically.
12. Are any traders based in Indore?

M011: Java Annotations and Reflection

Day 11:Java Annotations and Reflections API

In this module , you will understand the understand the java annotation and create custom annotations and explore about Reflection API.

Day Wise Schedule

- M0011: Java Annotation and Reflection API
- Day 11
  - Java Annotations
    - Built-In Java Annotations
    - Understanding Built-In Annotations in java
  - Java Custom Annotation
    - Marker Annotation
    - Single-Value Annotation
    - Multi-Value Annotation
  - Reflection in Java
    - Introduction to Reflection
    - java.lang.Class
    - Commonly used methods of Class class

Useful References

- Java Annotations
- https://www.javatpoint.com/java-annotation

- Custom Annotation
- https://www.javatpoint.com/custom-annotation

- Reflection
- https://www.javatpoint.com/java-reflection

M012:Java Database connection

Day 12:Java Database Connection

In this module , you will be able to connect to MySQL Database with Java, Handle SQL parameters with Prepared Statements, submit SQL statements to insert, update and delete data ,call stored procedure and handle various parameter types(IN, INOUT etc..)

Day Wise Schedule

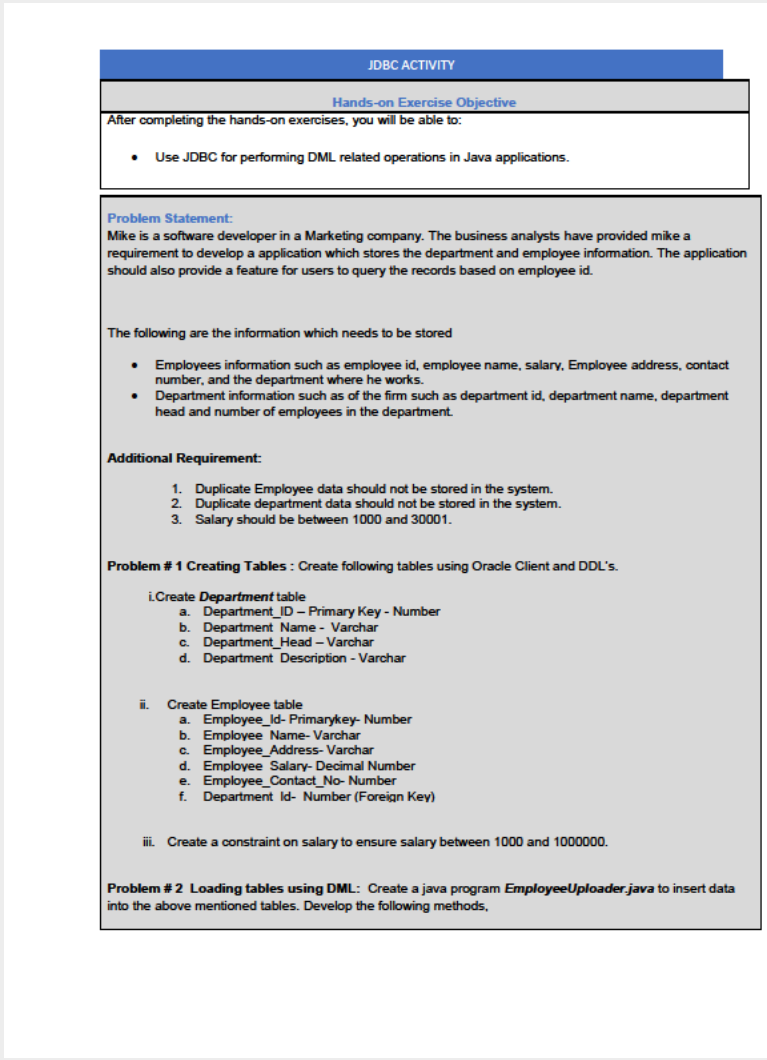
- M0012: Java Database Connections
- Day 12
  - Introduction to JDBC
    - Overview to Connection Interface
    - Working with Type 4 Driver
    - Overview to DriverManager class
    - Overview to Statement & PreparedStatement
    - Overview to ResultSet
  - Introduction to Layered Structure using JDBC
    - Conceptualizing & Implementing Connection Factory
    - Understanding & Implementing DAO classes
    - Using properties files for the database configuration.
    - Working on BankAppliation case study.

Useful References

- http://tutorials.jenkov.com/jdbc/index.html
- https://www.journaldev.com/2681/jdbc-tutorial

Advanced JDBC Assignment

double click on image



JDBC ACTIVITY

Hands-on Exercise Objective

After completing the hands-on exercises, you will be able to:

- Use JDBC for performing DML related operations in Java applications.

Problem Statement:

Mike is a software developer in a Marketing company. The business analysts have provided mike a requirement to develop a application which stores the department and employee information. The application should also provide a feature for users to query the records based on employee id.

The following are the information which needs to be stored

- Employees information such as employee id, employee name, salary, Employee address, contact number, and the department where he works.
- Department information such as of the firm such as department id, department name, department head and number of employees in the department.

Additional Requirement:

1. Duplicate Employee data should not be stored in the system.
2. Duplicate department data should not be stored in the system.
3. Salary should be between 1000 and 30001.

Problem # 1 Creating Tables : Create following tables using Oracle Client and DDL's.

i.Create ***Department*** table
   a. Department_ID – Primary Key - Number
   b. Department Name - Varchar
   c. Department_Head – Varchar
   d. Department Description - Varchar

ii. Create Employee table
   a. Employee_Id- Primarykey- Number
   b. Employee Name- Varchar
   c. Employee_Address- Varchar
   d. Employee Salary- Decimal Number
   e. Employee_Contact_No- Number
   f. Department Id- Number (Foreign Key)

iii. Create a constraint on salary to ensure salary between 1000 and 1000000.

Problem # 2 Loading tables using DML: Create a java program ***EmployeeUploader.java*** to insert data into the above mentioned tables. Develop the following methods,

M013 and 14:Mockito Framework

Day 13 and 14:Mockito Framework

In this module , you will learn what mocking is and why we should mock , learn to use Mockito stubbing to interact with mock objects, learn to use Mockito verification to confirm method calls ,learn to use stubbing and verification together to develop fully tested code and use powerMock

Day Wise Schedule

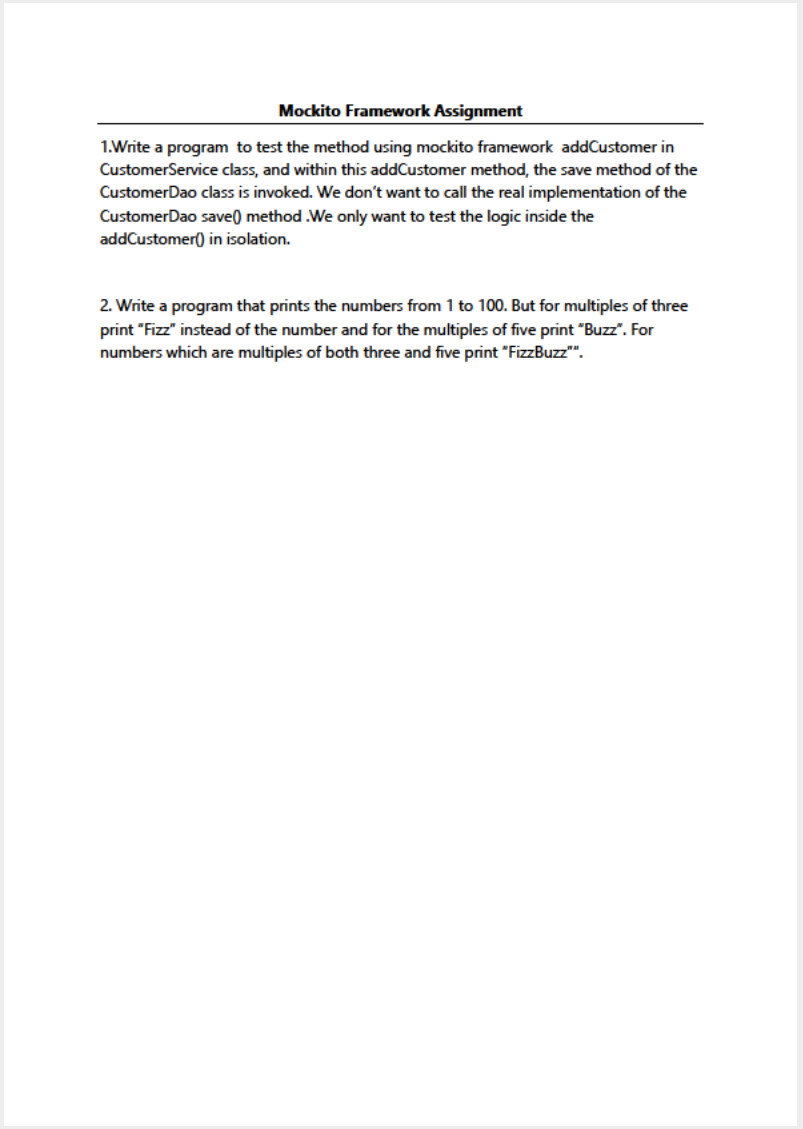
- M0013 and 14: Mockito Framework
- Day 13 & 14:
  - Working with Mockito
    - Testing with mock objects
    - Adding Mockito as dependencies to a project
    - Using the Mockito API
    - Exercise: Write an instrumented unit test using Mockito
    - Exercise: Creating mock objects using Mockito
    - Using PowerMock with Mockito
    - Using a wrapper instead of Powermock

Useful References

- Mockito tutorial
- https://www.vogella.com/tutorials/Mockito/article.html
- https://www.tutorialspoint.com/mockito/

Advanced Mockito Framework Assignment

double click on image



Mockito Framework Assignment

1.Write a program to test the method using mockito framework addCustomer in CustomerService class, and within this addCustomer method, the save method of the CustomerDao class is invoked. We don't want to call the real implementation of the CustomerDao save() method .We only want to test the logic inside the addCustomer() in isolation.

2. Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"".

Case Study

double click on image

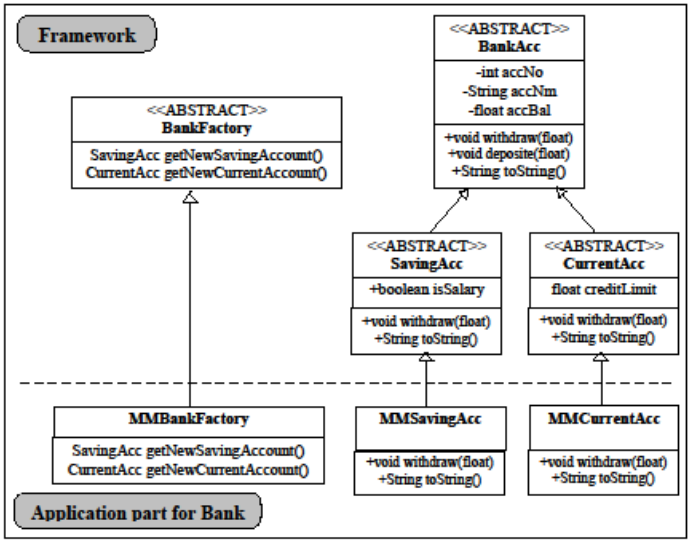
Java Case Study – I
Framework for Bank

Design a simple framework for Bank Application to represent Saving Accounts and Current Accounts.

Use the framework to design application for *MoneyMoney Bank*.

Objectives

- To understand the concept of framework in application development.
- Areas of application for Abstract classes, abstract methods etc.
- Polymorphism and its uses,
- Final fields and Lazy Initialization
- Getter and Setter methods
- Lazy Binding of methods



Thank you for your Participation

kindly reach p-v.sasirekha@capgemini.com for queries.