Core Java

1. Write a program to add, subtract, multiply and divide two non-zero hard-coded numbers.
2. Write a program to search for the greatest of three numbers using Short-circuit Operators and print the result.
3. Write a program –declare two variables a & b and initialize them to true and false respectively. Get the output of the following computations:

!a

a | b

(!a & b) | (a & !b)

Syntax

1. Write a program to add, subtract, multiply and divide two numbers using methods with parameters using only one class.
2. In one project, create two classes. One class should contain only methods (add, subtract, multiply and divide). The other class should contain only the main() method which calls each of the methods from the previous class.
3. Create an example for each control statement.

Class Members

1. Create a Customer class with:
   1. At least 2 constructors
   2. At least 2 overloaded methods
   3. At least 1 static variable
   4. At least 2 static methods
   5. At least 1 final variable
2. Create a main method in a new class that:
   1. Creates 2 customers
   2. Uses each of the Customer class members

OOP

1. Create an Animal interface with two methods: eat() and move(). Write a class called Mammal that implements Animal and the eat() but does not implement move(). Mammal should also have a static method called walk(). Create a Human class that extends Mammal, overrides eat() and has a static method called walk(). Create two objects as follows:

Animal one = new Human(); Mammal two = new Human(); Human three = new Human();

Call each of the methods for each of the objects: eat(), move(), walk(). Note the results.

Exceptions

1. Create a custom exception class that overrides the getMessage( ) method
2. Create a class that manually throws an exception of your custom type
3. Use System.out.println( ) to note where you are in the control flow. Example: “Starting try block”, “Ending try block”, “Starting catch block”, etc.
4. Duck the exception at least once
5. Implement a finally block that prints a graceful goodbye message
6. Use the System.exit(0) command in the try block and rerun the application. Note the console output to see if the finally block executes.

Advanced Concepts

1. Create a StringBuilder object. Use at least three methods to manipulate the String.
2. Create a new String with delimited tokens, such as “pickles:ketchup:mustard:onion” and use StringTokenizer to parse out and print each token.
3. Create two String objects with number values (i.e. “20). Write a method that adds the two.
4. Request garbage collection in your method.
5. Create a Runtime object and note at least three methods. Imagine how you would use them.

Collections

1. Create an ArrayList, LinkedList, HashMap, TreeSet, and a HashSet. Insert 3 objects into each.
2. Iterate over each collection and print each object.
3. Sort the collections and compare the original to the sorted collection.

File I/O

1. Consider a text file has the following colon-separated lines:
2. Employee id:First Name:Last Name:Role
3. Read the file, and parse(tokenize) the fields using StringTokenizer with delimiter(:)
4. While parsing the file, place the information in an object, which should have four instance variables: Employee Id, First Name, Last Name, Role.
5. Place these objects in a HashMap with the key as employee id and value as the object

i.e HashMap should have

Key: 01 Value : Object representing first line

Key: 02 Value: Object representing second line

Multithreading

1. Write a method that stores an integer. Each time the method is called, increment the integer by 1. Create three threads that call this method 1000 times each, storing the returned result in their own set. Print out each thread’s set.
2. Synchronize the method that stores and increments the integer. Execute your threads again.
3. Review thread-specific methods: sleep(), wait(), yield(), suspend(), stop(), resume(), interrupt(), notify(), notifyAll()

Easy Coding Exercises

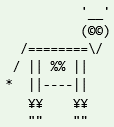
1. Write a program called CheckPassFail which prints "PASS" if the int variable "mark" is more than or equal to 50; or prints "FAIL" otherwise.
2. Write a program called CheckOddEven which prints "Odd Number" if the int variable “number” is odd, or “Even Number” otherwise. Hint: you may use modulus operator.
3. Write a program called PrintNumberInWord which prints "ONE", "TWO",... , "NINE", "OTHER" if the int variable "number" is 1, 2,... , 9, or other, respectively. Use (a) a "nested-if" statement; (b) a "switch-case" statement.
4. Write a program called KeyboardScanner to prompt user for an int, a double, and a String. Print the input onto the console.
5. Write a program called Arithmetic that takes three command-line arguments: two integers followed by an arithmetic operator (+, -, \* or /). The program shall perform the corresponding operation on the two integers and print the result.
6. Write a program called GradesStatistics, which reads in n grades (of int between 0 and 100, inclusive) and displays the average, minimum, maximum, and standard deviation. Your program shall check for valid input. You should keep the grades in an int[] and use a method for each of the computations.

Advanced Coding Exercises

1. Write a method to replace a given character in a String with another character.
2. Write a method that takes a date of birth via Calendar object and returns your age. If it is your birthday, print the Happy Birthday song!
3. Write a Java program to print the first 100 Fibonacci numbers using iteration. Rewrite using recursion.
4. Write a Java program that prints the numbers from 1 to 50. 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"
5. Write Comparators to compare two employees based on their name, department, and age.
6. Write a Java program to check if a number is Armstrong or not.
7. Write a method to reverse a String without using API functions.
8. Write a method to determine if a number is prime.

Bonus Exercises

1. Write a method that swaps two integer variables without using a third variable.
2. Create a Java application that finds the middle node of a LinkedList in one pass.
3. Write a Java application that computes the factorial of a number.
4. Implement the substring method.
5. Write a program that creates deadlock between two threads.
6. Write a program that creates livelock between two threads.
7. Write a program to reverse a number.
8. Find a duplicate number from an array of 1 to N numbers.
9. Write a program that counts repeated words in a file.
10. Write a program to find duplicate characters in a String.
11. Write a method that determines if a number is even or odd without using modulus.
12. Write a method that determines if a String is a palindrome.
13. Write a method that sorts an integer array from least to greatest without using API functions.
14. Write a program to find common elements between two arrays.
15. Write a program to implement hashcode and equals for an Employee object.
16. Write a Java application that gets distinct elements from an array—avoiding duplicate elements.
17. Write a program that gets all distinct words from a file.
18. Write a program called ComputePI to compute the value of π. JDK maintains the value of π in a double constant called Math.PI. Compare the values obtained and the Math.PI, in percents of Math.PI.
19. Write an application that prints the following pattern:



1. Write a Java application that prints Pascal’s Triangle.
2. Write a program called NumberGuess to play the number guessing game. The program shall generate a random number between 0 and 99. The player inputs his/her guess, and the program shall response with "Try higher", "Try lower" or "You got it in n trials" accordingly.
3. Write a Java console application for the classic Hangman game. Allow on player to enter a word that the other player attempts to guess (use Console.readPassword to obscure the text). For each incorrect answer, draw a piece of the Hangman on the console. Show a tally of all previously selected letters too.