Value Added Course on “Java Application Development”

Course Code: NCG31/CS02

Duration: 60 Hrs(Odd/Even Saturdays of even semester 2014-15).

Eligibility

The II Year B.E./B.Tech students admitted in the year 2013 and later. They can avail this course certificate to earn one credit for Group-III under Non-CGPA courses.

Course Overview

This value added course on “Java Application Development” aims to teach and train the aspiring Java programmers. The goal is to accelerate the programming skills they need for developingapplications in Javasuccessfully.

Course Objectives

- To explore the fundamentals of Java programming and how to use Java to develop applications
- To uncover core object-oriented concepts, including classes, packages, objects, methods, properties, abstraction, polymorphism, inheritance, encapsulation, and more

Course Outcomes

Upon successful completion of this course, the student will be able to:

- Explore the use of Eclipse as a Java IDE.
- Debug programs using a variety of methods including breakpoints, traces, and code stepping.
- Build Java programs that conform to Java coding style, naming, and documentation conventions.
- Demonstrate advanced object oriented development skills through the use of design patterns.
- Develop an understanding of the fundamental classes of the Java application framework.
- Use the Java collection framework and serialization to manage and save object data.
- Understand Java and its relationship to databases.
- Develop database applications using JDBC protocols.
- Build robust code using exception handling.

Content Delivery

A batch of 30 students will be provided with private, customized, on-site training at our premises. On content delivery side, 50% of industrial experts hands-on-practices and remaining 50% by in-house technical expertise.

Prerequisites

This course can be tailored for audiences ranging from first-time programmers to experienced OO developers seeking to learn Java. It is strongly recommended that all course attendees have similar levels of experience.

Hands-on/Lecture Ratio

This course is 60% hands-on, 40% lecture, with the longest lecture segments lasting for 40 minutes.

Case Studies for Java Fundamentals

1. **Number Game**

   Given three digits as input, create a 4 digit number out of each input in which all the digits are the same. Then add all the 3 numbers and return the value.

   add4DigitNums(1,2,3) = 6666 (as 1 will form 1111, 2 will form 2222, 3 will form 3333 and 1111+2222+3333 = 6666)
   add4DigitNums(4,5,6) = 16665 (as 4 will form 4444, 5 will form 5555, 6 will form 6666 and 4444+5555+6666 = 16665)

2. **Has scored a century**

   The scores of a batsman in his last three innings have been provided. You have to determine whether he has scored a century in the last three innings or not. If yes, return true else return false.
   isCentury(48,102,89) = true
   isCentury(99,46,81) = false

3. **Lottery prize**

   Jack bought a lottery ticket. He will get a reward based on the number of the lottery ticket. The rules are as follows
   If the ticket number is divisible by 4, he gets 6
   If the ticket number is divisible by 7, he gets 10
   If the ticket number is divisible by both 4 and 7, he gets 20
   Otherwise, he gets 0.

   Given the number of the lottery ticket as input, compute the reward he will receive
   lotteryReward(22) = 0

lotteryReward(16) = 6
lotteryReward(21) = 10
lotteryReward(56) = 20

4. **All primes between**

Given two numbers n1 and n2 as input, return an array containing all the primes between n1 and n2 (Note that both n1 and n2 can be included in the array if they are prime). Also, the primes in the array need to be in ascending order.

getPrimes(967,1041)={967,971,977,983,991,997,1009,1013,1019,1021,1031,1033,1039}
getPrimes(6,11)={7,11}

5. **`CozaLozaWoza`** **(Loop & Condition)**

Write a program called **CozaLozaWoza** which prints the numbers 1 to 110, 11 numbers per line. The program shall print "Coza" in place of the numbers which are multiples of 3, "Loza" for multiples of 5, "Woza" for multiples of 7, "CozaLoza" for multiples of 3 and 5, and so on. The output shall look like:

```
1 2 Coza 4 Loza Coza Woza 8 Coza Loza 11

Coza 13 Woza CozaLoza 16 17 Coza 19 Loza CozaWoza 22

23 Coza Loza 26 Coza Woza 29 CozaLoza 31 32 Coza
```

6. **`PhoneKeyPad`**

On your phone keypad, the alphabets are mapped to digits as follows: `ABC(2)`, `DEF(3)`, `GHI(4)`, `JKL(5)`, `MNO(6)`, `PQRS(7)`, `TUV(8)`, `WXYZ(9)`.Write a program called `PhoneKeyPad`, which prompts user for a `String` (case insensitive), and converts to a sequence of Keypad digits. Use a nested-if (or switch-case) in this exercise. Modify your program to use an *array* for table look-up later.

Hints: You can use `in.next().toLowerCase()` to read a string and convert it to lowercase to reduce your cases.

7. **`TestPalindromicWord`**

A word that reads the same backward as forward is called a *palindrome*, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive). Write a program called `TestPalindromicWord`, that prompts user for a word and prints ""`xxx" is|is not a palindrome`".

Hints: Read in a word and convert to lowercase via `in.next().toLowercase()`.

7. **`NumberGuess`**

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.

For example:

```
> java NumberGuess
Key in your guess: 50
Try higher
70
Try lower
65
Try lower
You got it in 4 trials!
```

Hints: Use `Math.random()` to produce a random number in `double` between `0.0` and (less than) `1.0`. To produce an int between `0` and `99`, use:

```
int secretNumber = (int)(Math.random()*100);
```

8. WordGuess

Write a program called `WordGuess` to guess a word by trying to guess the individual characters. The word to be guessed shall be provided using the command-line argument. Your program shall look like:

```
> java WordGuess testing
Key in one character or your guess word: t
Trail 1: t__t___
Key in one character or your guess word: g
Trail 2: t__t__g
Key in one character or your guess word: e
Trail 3: te_t__g
Key in one character or your guess word: testing
Trail 4: Congratulation!
You got in 4 trials
```

Hints: Set up a `boolean` array to indicate the positions of the word that have been guessed correctly. Check the length of the input `String` to determine whether the player enters a single character or a guessed word. If the player enters a single character, check it against the word to be guessed, and update the `boolean` array that keeping the result so far.

Try retrieving the word to be guessed from a text file (or a dictionary) randomly.

9. **Count chars in same position**

Given 2 strings, str1 and str2, as input, return the count of the chars which are in the same position in str1 and str2.

count(“Hello”,”World”)=1 (Because l is at 3rd position in both Hello and World)
count(“New York”,”New Delhi”)=4 (because “New ” are in the position

count(“rhinoceroses”,”hippopotamus”)=2 (because o is 4 position in both, s is in 11th position in both)

10. **Multiple Choice Questions**

You have been given two string arrays as input, key and answers. The “key” array contains the correct answers of an examination, like {“a”,”b”,”d”,”c”,”b”,”d”,”c”}. The “answers” array contains the answers that a student has given. You can assume that the student has answered all the questions. While scoring the examination, a correct answer gets +3 marks while an incorrect answer gets -1 marks. Calculate the score of the student.

studentScore({“a”,”b”,”d”,”c”,”b”,”d”,”c”},{“a”,”b”,”b”,”c”,”a”,”d”,”b”}) = 9 (4 are correct and 3 are wrong)
studentScore({“c”,”b”,”d”,”c”},{“a”,”d”,”b”,”a”}) = -4

Case Studies for Java Application Development

(Solve Any Four Choices)

1. **Tennis Match**

   Calculate the winning probabilities of the two players for a tennis match of any type. User can specify the number of sets of the match (i.e. 3 setter or 5 setter), and whether the final set uses a tie break at 6-6 or continue alternating serve games until a winner emerges. Note all the previous sets are assumed to use tie break. User also needs to provide the winning probability of each player's service game. The other inputs are the current status (game scores, set scores, and overall match scores). The program calculates the winning probability of the two players under Markovian assumption and constant service game winning probability for each player.

2. **Word ladder**

   Write a program WordLadder.java that takes two 5 letter strings as command-line arguments, and reads in a list of 5 letter words from standard input, and prints out a shortest (or word ladder connecting the two strings (if it exists). This word game was invented by Lewis Carroll and was orignally known as a doublet. Two words can be connected in a word ladder chain if they differ in exactly one letter. As an example, the following word ladder connects green and brown.

green	greet	great	groat	groan	grown	brown

3. **Create the classes Item and ShoppingCart**

   Create a class Item which refers to an item in the shopping cart of an online shopping site like Amazon. The class Item has the fields name (of type String), productId (of type String), price (of type double), quantity (of type int), amount (of type double). The field amount is calculated as price * quantity.

   Also define a class ShoppingCart which refers to the shopping cart of a customer at Amazon. The class ShoppingCart has the fields items (of type Item[ ]), totalAmount (of type double).

   Also define a class TestCart which has a function makeCart(String[ ] itemData) which takes the information about the shopping cart of a customer as input and returns an object of type ShoppingCart.The input String[] items, consists of an array of String where each String provides information on an item in the manner "name, id, price, quantity".

   For Example,

   It can be "Colgate,CP10023,54.50,3″ where Colgate is the name, CP10023 is the product id, 54.50 is the price and 3 is the quantity. Note that the ShoppingCart object returned should have the correct totalAmount and each Item in it should the correct amount.

4. Billing Counter

The problem here simulates the action of the billing counter of a department store.Two inputs have been provided as String[ ], master and billscan. The String[ ] master, contains the following details for each product : code, description, price and applicable discount. The String[ ] billscan consists of the codes being scanned at the billing counter. Note that billscan can have the same code appear multiple times.

The objective is to generate the itemized bill for the billscan provided. The itemized bill is an object of the class ItemizedBill and consists of a Vector of BillItems and the total value of the bill. Each BillItem corresponds to the different products being purchased at the counter. The quantity purchased is reflected in the quantity field of the BillItem. You have been provided the following Java files in the download, Product, ProductMaster, BillItem, ItemizedBill and BillingCounter. Also draw the class diagram for this problem

5. Zoo Animals

Given a zoo, the objective is to compute the food requirements of the zoo as well create a count of various animals. All these values are to be stored in the object ZooAnimals. There is a function foodCalc(String[] animalDetails, String[] animalList) in the class ZooFoodSystem which needs to be defined. Note that while returning the object ZooAnimals, all the fields in the object should have the correct values. The inputs to the function foodCalc are

String[ ] animalDetails : Name, Food type (carnivore / herbivore) and amount of daily food consumption.

String[ ] animalList : List of all animals found in the zoo. An animal will occur as many times in the list as they are in the zoo.

6. Infinite precision Arthmetic

Make a class InfinitePrecision than can hold as much number of digits (in String Form) as possible. Design the constructor that can check the String that has only valid digits. Add methods to show particular digit, addition of two such numbers by overloading + operator, subtraction of two such numbers by overloading – operator and a comparator to return which return boolean true if first digit is bigger.

Hint: Refer BigInteger Class features in Java

7. **Abbreviation Dictionary**
   Develop an application that can store abbreviations and their meanings. Allow user to enter new abbreviations, change the meaning of the existing abbreviation and look-up the meaning of an existing abbreviation. Demonstrate this application with suitable selection of Data structures.

8. **Bank Accounts Management**
   A BankAccount class support the getBalance()method. They also support the deposit()and withdraw()methods of its customer. Implement CheckingAccount sub-Class. Each CheckingAccountobject has two instance fields:
   balance (inherited from BankAccount),
   transactionCount(new to CheckingAccount). You can apply four methods to CheckingAccount objects as,
   - getBalance()(inherited from BankAccount)
   - deposit(doubleamount)(overrides BankAccountmethod)
   - withdraw(doubleamount)(overrides BankAccountmethod)
   - deductFees()(new to CheckingAccount)

Also, Throw possible Exceptions to the customers such as InSufficientCredentials, NotAValidAmount

Training Materials

All registered students for course can receive a soft copy of *Course Guide which was prepared as per the guidelines of Industries*.

System Setup

- Core 2 Duo or faster processor with at least 3GB RAM
- Any operating system that supports Java 1.7 or later
- JDK 7 (6 could be accommodated upon request)
- The Java tool the students are likely to use after the class (Eclipse is recommended, but IBM RAD, NetBeans, and other tools are supported)

Assessment Procedure

- The course learner’s outcomes will be assessed through in-house written examination and online examinations.

Contact Person (for Registration)

Dr.B.Paramasivan, M.E., Ph.D., FIE, SMIEEE

Professor & Head / CSE Dept.,

National Engineering College (Autonomous),

K.R.Nagar, Kovilpatti – 628 503.