Core Java 8 Lab Book

Link to this document:
https://docs.google.com/document/d/1jsVLzZG5nsWCwYn2jICkMcnCrlsq_IC2XRElmjclnak/edit?usp=sharing

Table of Contents

Before you start:
Before your start developing the solutions here are some tips which can make your problem solving easier.

1. Always check if there are any direct API methods available to solve the question easily

2. Use Collection.sort method if you want to sort an arraylist. In case of sorting arrays convert the array and use Collection.sort method to sort it.

3. Converting the numeric data types to string may help to solve some problems for example if you are asked to check if the first digits of two numbers are same convert the two numbers to String and use the charAt() method to check it or if you want to reverse the digits of a number etc.

4. If you are asked to remove the duplicate elements in an array. Convert it to set object. If the array needs to be sorted order go for TreeSet

5. Try to use collection,string and wrapper APIs where ever possible.

6. While using any API methods just go through the other methods in the same API which may help you in solving other problems

7. The Hints provided are just to help the associate solve the problem in the best way. You can use your own algorithm/logic to solve the problem.

Getting Started

Overview

This lab book is a guided tour for learning Core Java version 8. It comprises of assignments to be done. Refer the demos and work out the assignments given by referring the case studies which will expose you to work with Java applications.

Setup Checklist for Core Java

Here is what is expected on your machine in order to work with lab assignment.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 7 or higher.
- Memory: (1GB or more recommended)
- Internet Explorer 9.0 or higher or Google Chrome 43 or higher

Please ensure that the following is done:

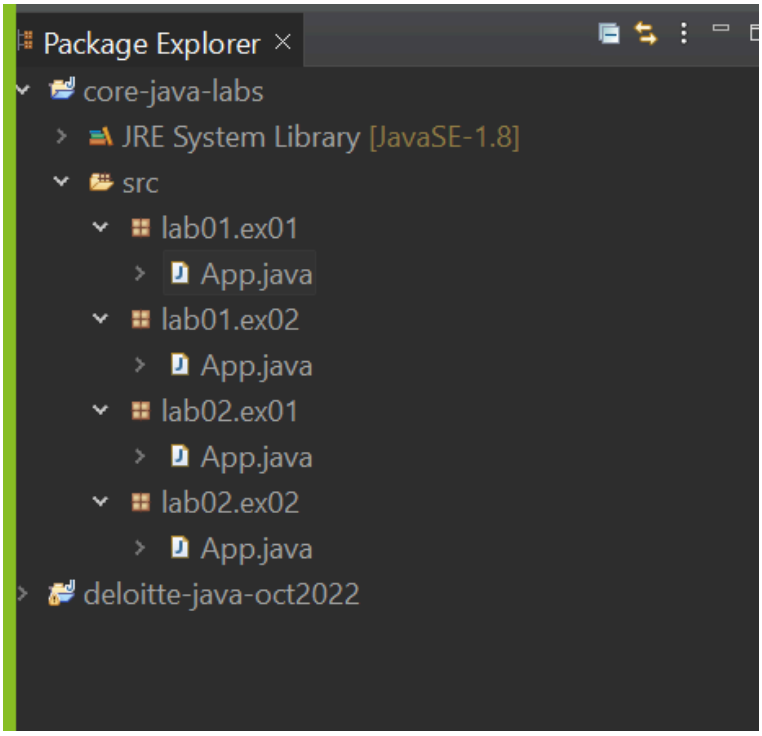
- A text editor like Notepad , Eclipse, STS is installed.
- JDK 1.8 or above is installed. (This path is henceforth referred as <java_home>)

Instructions

- For all Naming conventions, refer Appendix A. All lab assignments should adhere to naming conventions.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory java_assignments. For each lab exercise create a directory as lab <lab number>.

Learning More (Bibliography if applicable)

- https://docs.oracle.com/javase/8/docs/
- Java, The Complete Reference; by Herbert Schildt
- Thinking in Java; by Bruce Eckel
- Beginning Java 8 Fundamentals by KishoriSharan


Package Explorer
core-java-labs
JRE System Library [JavaSE-1.8]
src
lab01.ex01
App.java
lab01.ex02
App.java
lab02.ex01
App.java
lab02.ex02
App.java
deloitte-java-oct2022

Lab 1: Flow control/Operators/Assignments

Optional----------

Exercise 1: Create a method to find the sum of the cubes of the digits of an n digit number

Exercise 2: Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with int inputs. On entering the choice, an appropriate message with “stop” or “ready” or “go” should appear in the console.

Exercise 3: The Fibonacci sequence is defined by the following rule. The first 2 values in the sequence are 1, 1. Every subsequent value is the sum of the 2 values preceding it. Write a Java program that uses both recursive and non-recursive functions to print the nth value of the Fibonacci sequence?

Exercise 4: Write a Java program that prompts the user for an integer and then prints out all the prime numbers up to that Integer?

Mandatory----------

Exercise 5: Create a class with a method which can calculate the sum of first n natural numbers which are divisible by 3 or 5.

Note: Complete this exercise in Eclipse.[Eclipse-Lab1_1]

Method Name	calculateSum
Method Description	Calculate Sum
Argument	int n
Return Type	int-sum
Logic	Calculate the sum of first n natural numbers which are divisible by 3 or 5.

Exercise6: Create a class with a method to find the difference between the sum of the squares and the square of the sum of the first n natural numbers.

Note: Complete this exercise in Eclipse.[Eclipse-Lab1_2]

Method Name	calculateDifference
Method Description	Calculate the difference
Argument	int n
Return Type	int - Sum

Logic	Find the difference between the sum of the squares of the first n natural numbers and the square of their sum. For Example if n is 10,you have to find (1^2+2^2+3^2+….9^2+10^2)-(1+2+3+4+5…+9+10)^2

Exercise 7: Create a method to check if a number is an increasing number

Note: Complete this exercise in Eclipse.[Eclipse-Lab1_3]

Method Name	checkNumber
Method Description	Check if a number is an increasing number
Argument	int number
Return Type	boolean
Logic	A number is said to be an increasing number if no digit is exceeded by the digit to its left. For Example : 134468 is an increasing number

Exercise 8: Create a method to check if a number is a power of two or not

Note: Complete this exercise in Eclipse.[Eclipse-Lab1_4]

Method Name	checkNumber
Method Description	Checks if the entered number is a power of two or not
Argument	int n
Return Type	boolean
Logic	Check if the input is a power of two. Ex: 8 is a power of 2

Lab 2: Arrays

Mandatory----------

Exercise 1: Create a method which accepts an array of integer elements and return the second smallest element in the array

Note: Complete this exercise in Eclipse.[Eclipse-Lab3_1]

Method Name	getSecondSmallest
Method Description	Get the second smallest element in the array
Argument	int[]
Return Type	int
Logic	Sort the array and return the second smallest element in the array

Exercise 2: Create a method that can accept an array of String objects and sort in alphabetical order. The elements in the left half should be completely in uppercase and the elements in the right half should be completely in lower case. Return the resulting array.
Note: If there are odd number of String objects, then (n/2) +1 elements should be in UPPPERCASE

Note: Complete this exercise in Eclipse.[Eclipse-Lab3_2]

Method Name	sortStrings
Method Description	accept an array of String objects and sort in Alphabetical order.
Argument	String[] arr
Return Type	String
Logic	

Exercise 3: Create a method which accepts an integer array, reverse the numbers in the array and returns the resulting array in sorted order

Note: Complete this exercise in Eclipse.[Eclipse-Lab3_3]

Method Name	getSorted
Method Description	Return the resulting array after reversing the numbers and sorting it
Argument	int []
Return Type	int
Logic	Accept and integer array, reverse the numbers in the array, sort it and return the resulting array.

	Hint Convert the numbers to String to reverse it

Exercise 4: Create a method which accepts an integer array and removes all the duplicates in the array. Return the resulting array in descending order

Method Name	modifyArray
Method Description	Remove duplicates
Argument	int []
Return Type	int []
Logic	Remove the duplicate elements in the array and sort it in descending order

Lab 3: Strings and Parsing

Exercise 1: Write a Java program that reads a line of integers and then displays each integer and the sum of all integers. (Use String Tokenizer class)?

Note: Complete this exercise in Eclipse.[Eclipse-Lab6_1]

Exercise 2: Create a class containing a method to create the mirror image of a String. The method should return the two Strings separated with a pipe(|) symbol .

Method Name	getImage
Method Description	Generate the mirror image of a String and add it to the existing string.
Argument	String
Return Type	String
Logic	Accepts One String Find the mirror image of the String Add the two Strings together separated by a pipe(|) symbol. For Example Input : EARTH Output : EARTH|HTRAE Hint: Use StringBuffer API (Ex: For this problem reverse method in Stringbuffer can be used) Note: Learn the other APIs in StringBuffer

Exercise 3: Create a method which accepts a String and replaces all the consonants in the String with the next alphabet.
Note: Consonant refers to all alphabets excluding vowels

Method Name	alterString
Method Description	Replace consonants
Argument	String
Return Type	String
Logic	Return the String replacing all the consonants with the next character. For Example :JAVA should be changed as KAWA

Exercise 4: Create a method that accepts a number and modifies it such that the each of the digit in the newly formed number is equal to the difference between two consecutive digits in the original number. The digit in the units place can be left as it is.
Note: Take the absolute value of the difference. Ex: 6-8 = 2

Method Name	modifyNumber
Method Description	Accepts a number and modify it as per the requirement
Argument	int number1
Return Type	int
Logic	Accept a number and modify it such that the each of the digit in the newly formed number is equal to the difference between two consecutive digits in the original number. For example. Input: 45862 Output:13242 **Algorithm:** 🎬 Convert number into String 🎬 Extract each char using charAt method 🎬 Convert char to int and find the difference 🎬 Create new StringBuffer object and keep adding the difference 🎬 Finally convert StringBuffer to int

Exercise 5: Write a Java program that displays the number of characters, lines and words in a text?

Exercise 8: Create a method that accepts a String and checks if it is a positive string. A string is considered a positive string, if on moving from left to right each character in the String comes after the previous characters in the Alphabetical order. For Example: ANT is a positive String (Since T comes after N and N comes after A). The method should return true if the entered string is positive.
Note: Complete this exercise in Eclipse.[Eclipse-Lab6_5]

Exercise 9: Create a method to accept date and print the duration in days, months and years with regards to current system date.

Lab 4: Inheritance and Polymorphism

----------Optional----------

Exercise1: Create Person and Account Class as shown below in class diagram. Ensure minimum balance of INR 500 in a bank account is available.

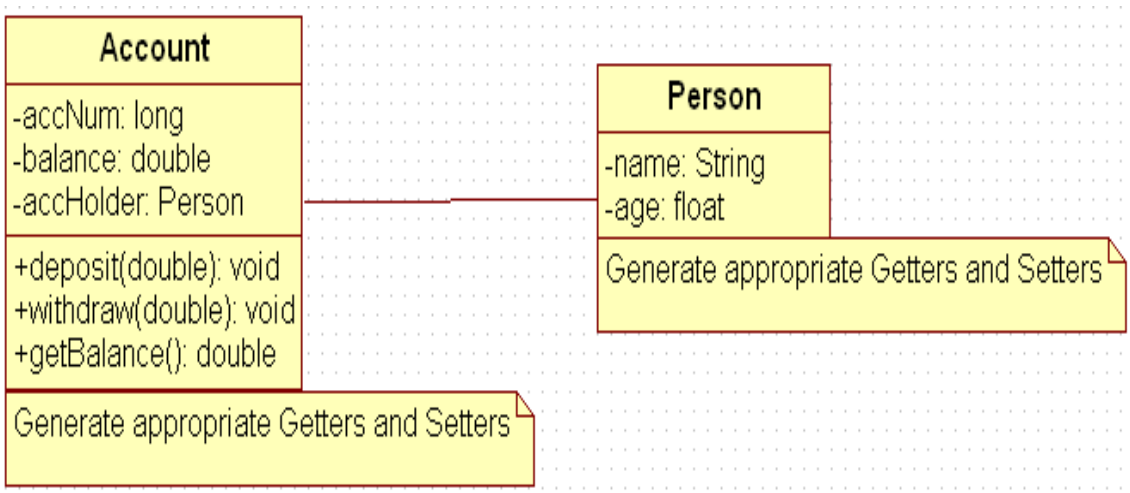


Figure 14: Association of person with account class

a) Create Account for smith with initial balance as INR 2000 and for Kathy with initial balance as 3000.(accNum should be auto generated).
b) Deposit 2000 INR to smith account.
c) Withdraw 2000 INR from Kathy account.
d) Display updated balances in both the account.
e) Extend the functionality through Inheritance and polymorphism. Inherit two classes Savings Account and Current Account from account class. And Implement the following in the respective classes.

- **Savings Account**

a. Add a variable called minimum Balance and assign final modifier.
b. Override method called withdraw (This method should check for minimum balance and allow withdraw to happen)

- **Current Account**

a. Add a variable called overdraft Limit
b. Overridemethod called withdraw (checks whether overdraft limit is reached and returns a Boolean value accordingly)

Exercise 2: create packages and classes as given below:

a) com.cg.eis.bean

In this package, create "Employee" class with different attributes such as id, name, salary, designation, insuranceScheme.

b) com.cg.eis.service

This package will contain code for services offered in Employee Insurance System. The service class will have one EmployeeService Interface and its corresponding implementation class.

The services offered by this class are:

i) Get employee details from user.

ii) Find the insurance scheme for an employee based on salary and designation.

iii) Display all the details of an employee.

c) com.cg.eis.pl

This package will contain code for getting input from user, produce expected output to the user and invoke services offered by the system.

----------**Mandatory**----------

Exercise 3: Using an inheritance hierarchy, design a Java program to model items at a library (books, journal articles, videos and CDs.) Have an abstract superclass called Item and include common information that the library must have for every item (such as unique identification number, title, and number of copies). No actual objects of type Item will be created - each actual item will be an object of a (non-abstract) subclass. Place item-type-specific behavior in subclasses (such as a video's year of release, a CD's musical genre, or a book's author).
More in detail:

1. Implement an abstract superclass called Item and define all common operations on this class (constructors, getters, setters, equals, toString, print, checkIn, checkOut, addItem, etc). Have private data for: identification number, title, and number of copies.

2. Implement an abstract subclass of Item named WrittenItem and define all common operations on this class. Added private data for author.

3. Implement 2 subclasses of WrittenItem: Book and JournalPaper.

3.1. Class Book: no new private data. When needed, override/overload methods from the superclass.
3.2. Class JournalPaper: added private data for year published. When needed, override/overload methods from the superclass.

4. Implement another abstract subclass of Item named MediaItem and define all common operations on this class. Added private data for runtime (integer).

5. Implement 2 subclasses of MediaItem: Video and CD.

5.1. Class Video: added private data for director, genre and year released. When needed, override/overload methods from the superclass.
5.2. Class CD: added private data for artist and genre. When needed, override/overload methods from the superclass.

Write the definitions of these classes and a client program (your choice!) showing them in use.

Note: Complete this exercise in Eclipse.

Lab 5: Exception Handling

Exercise 1: Validate the age of a person and display proper message by using user defined exception. Age of a person should be above 15.

Note: Complete this exercise in Eclipse.Eclipse Age Exception

Exercise 2: Write a Java Program to validate the full name of an employee. Create and throw a user defined exception if firstName and lastName is blank.

Note: Complete this exercise in Eclipse.Eclipse Name Exception

Exercise 3: Create an Exception class named as "EmployeeException"(User defined Exception) in a package named as "com.cg.eis.exception" and throw an exception if salary of an employee is below than 3000. Use Exception Handling mechanism to handle exception properly.

Lab 6: Collection Framework

----------**Mandatory**----------

Exercise 1: Create a method which accepts a hash map and return the values of the map in sorted order as a List.

Method Name	getValues
Method Description	Get the values of a map in sorted order
Argument	HashMap
Return Type	List
Logic	Return the values of a hash map in sorted order

Note: Complete this exercise in Eclipse.[Eclipse-Lab7_1]

Exercise2 Collection: Create a method that accepts a character array and count the number of times each character is present in the array.

Note: Complete this exercise in Eclipse.[Eclipse-Lab3_4]

Method Name	countChars
Method Description	method that accepts a character array and count the number of times each character is present in the array.
Argument	char[] arr
Return Type	Map<Character, Integer>
Logic	

Exercise 3: Create a method which accepts an array of numbers and returns the numbers and their squares in Hashmap

Method Name	getSquares
Method Description	Accepts a list of numbers and return their squares
Argument	int[]
Return Type	Map
Logic	Iterate through the list, find the square of each number and add the elements to a map object with the number as the key and the square as the value

Note: Complete this exercise in Eclipse.[Eclipse-Lab7_3]

Exercise 4: school offers medals to the students of tenth based on the following criteria

If(Marks>=90) : Gold
If(Marks between 80 and 90) : Silver
If(Marks between 70 and 80) : Bronze

Note: Marks between 80 and 90 means ☾ marks>=80 and marks<90

Write a function which accepts the marks of students as a Hashmap and return the details of the students eligible for the medals along with type of medal.
The input Hashmap contains the student registration number as key and mark as value.
The output Hashmap should contain the student registration number as key and the medal type as value.

Method Name	getStudents
Method Description	Generate the list of students eligible for scholarship
Argument	Hashmap
Return Type	Hashmap
Logic	The method should return the details of the students eligible for the medals along with the medal type.

Exercise 5: Create a method which accepts an array of integer elements and return the second smallest element in the array

Method Name		getSecondSmallest
Method Description		Get the second smallest element in the array
Argument		int[]
Return Type		int
Logic		Sort the array and return the second smallest element in the array Hint: 1. Convert to Array List 2. Use sort method in Collections class

Exercise 6: Create a method which accepts the id and the age of people as a Map and decide if they are eligible for vote. A person is eligible for vote if his age is greater than 18. Add the IDs of all the eligible persons to list and return the list.

Method Name	votersList
Method Description	Generate the list of voters based on the ages of the people
Argument	Map
Return Type	List
Logic	Accept a map with ID as key and Date of Birth as value and check if the person is eligible to vote. A person is eligible for vote for if his age is greater than 18. If the person is eligible add his ID to the list.

Exercise 7: Create a method which accepts an integer array, reverse the numbers in the array and returns the resulting array in sorted order

Method Name	getSorted
Method Description	Return the resulting array after reversing the numbers and sorting it
Argument	int []
Return Type	int
Logic	Accept and integer array, reverse the numbers in the array, sort it and return the resulting array. Hint 1. Convert the numbers to String to reverse it 2. Use Collection APIs to sort it

Lab 7: Layered Architecture

Refer the case study and create an application for that requirement by creating packages and classes as given below:

Case Study:

1. **Employee Medical Insurance Scheme:**

- By default, all employees in an organization will be assigned with a medical insurance scheme based on the salary range and designation of the employee.

  Refer the below given table to find the eligible insurance scheme specific to an employee.

Salary	Designation	Insurance scheme
>5000 and < 20000	System Associate	Scheme C
>=20000 and <40000	Programmer	Scheme B
>=40000	Manager	Scheme A
<5000	Clerk	No Scheme

- **com.cg.eis.bean**

  In this package, create “Employee” class with different attributes such as id, name, salary, designation, insuranceScheme.

- **com.cg.eis.service**

  This package will contain code for services offered in Employee Insurance System. The service class will have one EmployeeService Interface and its corresponding implementation class.

- **com.cg.eis.pl**

  This package will contain code for getting input from user, produce expected output to the user and invoke services offered by the system.

On the basis of above case study implement a class to accept multiple employee details and store all employee objects in a Hashmap. The functionalities need to be implemented are as follows:

- Add employee details to Hashmap.
- Accept insurance scheme from user and display employee details based on Insurance scheme.
- Delete an employee details from map.
- Build Expectation

Note: Complete this exercise in Eclipse.[Eclipse-Lab11]

Lab 8: Multithreading

Exercise 1: Write a program to do the following operations using Thread:

- Create an user defined Thread class called as “CopyDataThread .java” .
- This class will be designed to copy the content from one file “source.txt ” to another file “target.txt” and after every 10 characters copied, “10 characters are copied” message will be shown to user.(Keep delay of 5 seconds after every 10 characters read.)
- Create another class “FileProgram.java” which will create above thread. Pass required File Stream classes to CopyDataThread constructor and implement the above functionality.

Exercise 2: Write a thread program to display timer where timer will get refresh after every 10seconds.(Use Runnable implementation)

Lab 09: Concurrent Patterns in Java

Exercise 1: Implement the Multithreading Assignments using Executor, ExecutorService interface.

Exercise 2: You are asked to create an application for registering the details of jobseeker. The requirement is:
Username should always end with _job and there should be at least minimum of 8 characters to the left of _job. Write a function to validate the same. Return true in case the validation is passed. In case of validation failure return false.

Lab 9: Lambda Expressions and Stream API -Optional

Exercise 1: Write a lambda expression which accepts x and y numbers and return x^y.

Exercise 2: Write a method that uses lambda expression to format a given string, where a space is inserted between each character of string. For ex., if input is "CG", then expected output is "C G".

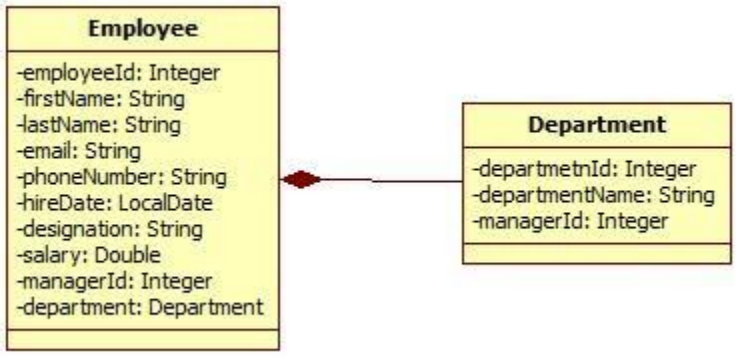
Exercise 3: Write a method that uses lambda expression to accept username and password and return true or false. (**Hint:** Use any custom values for username and password for authentication)

Exercise 4: Write a class with main method to demonstrate instance creation using method reference. (**Hint:** Create any simple class with attributes and getters and setters)

Exercise 5: Write a method to calculate factorial of a number. Test this method using method reference feature.

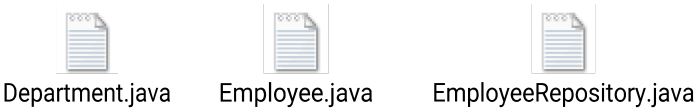
Case Study for Steam API:

Refer the classes given below to represent employees and their departments.



Class Diagram used for Stream API

Also refer an EmployeeRepository class which is used to create and populate employee's collection with sample data.



Create an EmployeeService class which queries on collections provided by EmployeeRepository class for following requirements. Create separate method for each requirement. (**Note:** Each requirement stated below must be attempted by using lambda expressions/stream API. It's

mandatory to solve at least 5 questions from following set. However, it is recommended to solve all questions to understand stream API thoroughly).

- Find out the sum of salary of all employees.
- List out department names and count of employees in each department.
- Find out the senior most employee of an organization.
- List employee name and duration of their service in months and days.
- Find out employees without department.
- Find out department without employees.
- Find departments with highest count of employees.
- List employee name, hire date and day of week on which employee has started.
- List employee name, hire date and day of week for employee started on Friday. (Hint: Accept the day name for e.g. FRIDAY and list all employees joined on Friday)
- List employee's names and name of manager to whom he/she reports. Create a report in format "employee name reports to manager name".
- List employee name, salary and salary increased by 15%.
- Find employees who didn't report to anyone (**Hint:** Employees without manager)
- Create a method to accept first name and last name of manager to print name of all his/her subordinates.
- Sort employees by their
    - Employee id
    - Department id
    - First name

Lab 10: File Handling - Optional

----------File Handling----------

Exercise 1: Write a Java program that reads a file and displays the file on the screen, with a line number before each line?

Exercise 2: Write a Java program that reads on file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes?

Appendices

Appendix A: Naming Conventions

Package names are written in all lower case to avoid conflict with the names of classes or interfaces.Companies use their reversed Internet domain name to begin their package names—for example, com.cg.mypackage for a package named mypackage created by a programmer at cg.com.

Packages in the Java language itself begin with **java. Or javax**.

Classes and interfaces The first letter should be capitalized, and if several words are linked together to form the name, the first letter of the inner words should be uppercase (a format that's sometimes called "camelCase").

For classes, the names should typically be nouns. For example:

Dog

Account

PrintWriter

For interfaces, the names should typically be adjectives like

Runnable

Serializable

Methods The first letter should be lowercase, and then normal camelCase rules should be used. In addition, the names should typically be verb-noun pairs. For example:

getBalance

doCalculation

setCustomerName

Variables Like methods, the camelCase format should be used, starting with a lowercase letter. Sun recommends short, meaningful names, which sounds good to us. Some examples:

buttonWidth

accountBalance

myString

Constants Java constants are created by marking variables static and final. They should be named using uppercase letters with underscore characters as separators:

MIN_HEIGHT