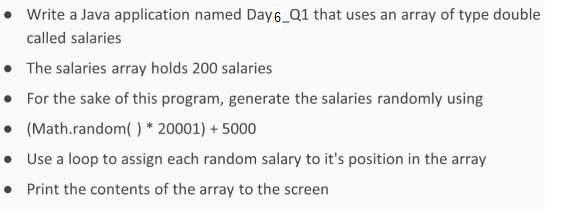
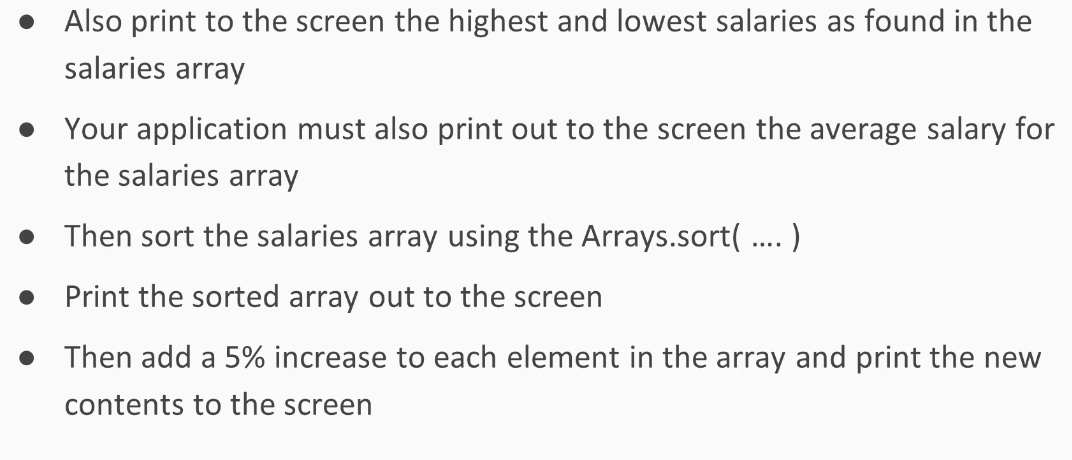
Hands On Exercises - Collections and Regular Expressions

**Setup Instructions:**

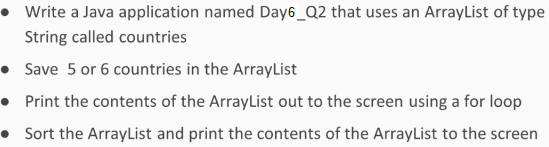
1. Create a new Java Project called “FullStackDay6”.
2. Create your classes in the package “collections”.
3. Define your classes as given below for each assignment.

Assignment 01

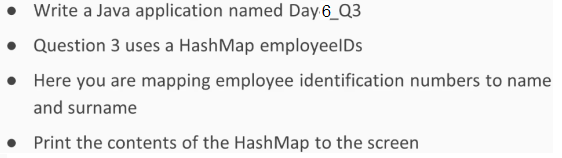


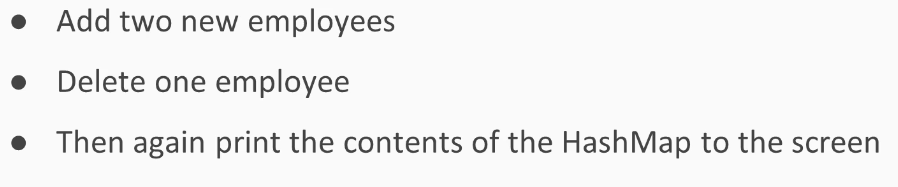


Assignment 02



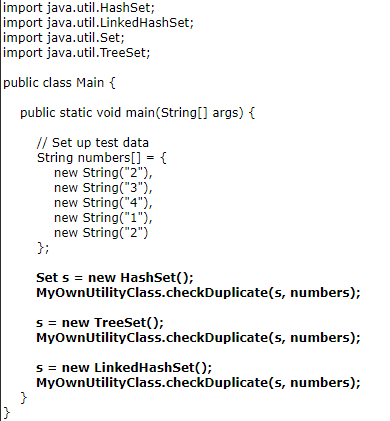
Assignment 03



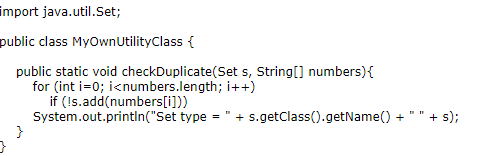


Assignment 04

* Write a Java application as follows.
* “setpolymorphism” is the sub-package name under package “collections”.
* Create a class ”Main” with a main method as follows:



* Write MyOwnUtilityClass.java.



* Build and run the project

**Output:**



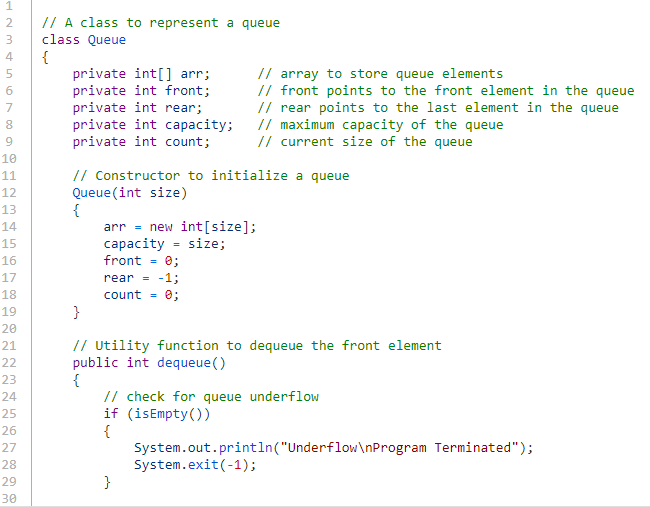
Assignment 05

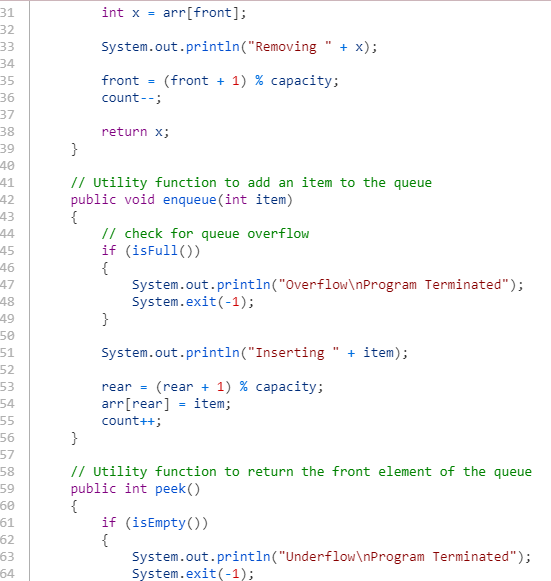
The next two assignments cover queue implementation in Java. A queue is a linear data structure that follows the FIFO (First–In, First–Out) principle. That means the object inserted first will be the first one out, followed by the object inserted next.

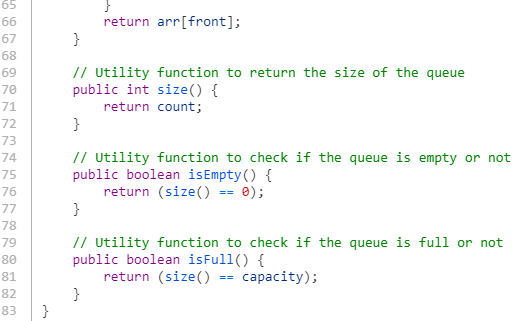
The queue supports the following core operations:

1. Enqueue: Inserts an item at the rear of the queue.
2. Dequeue: Removes the object from the front of the queue and returns it, thereby decrementing queue size by one.
3. Peek: Returns the object at the front of the queue without removing it.
4. IsEmpty: Tests if the queue is empty or not.
5. Size: Returns the total number of elements present in the queue.

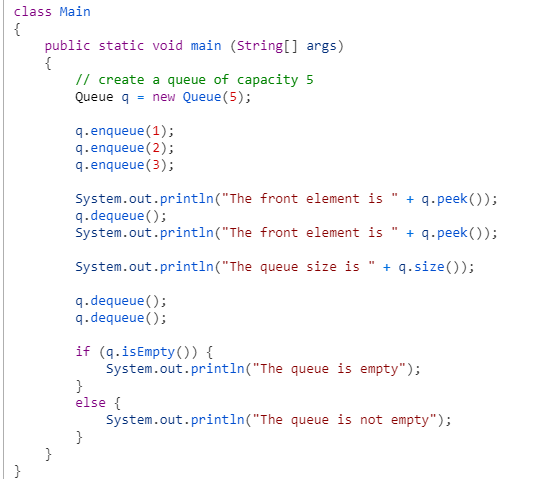
* Write a Java application named Queue.
* “queuewitharray” is the sub-package name under package “collections”.
* Create the following class :

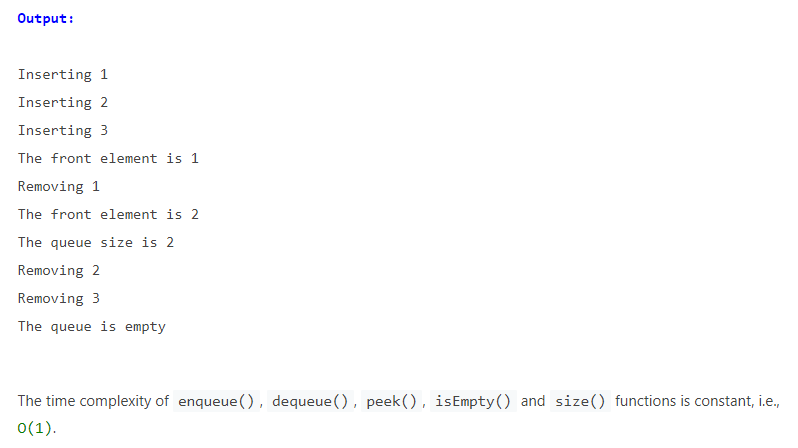






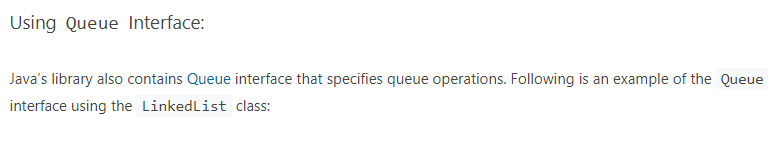
* Create a class ”Main” with a main method as follows:



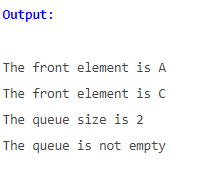


Assignment 06

* Write a Java application as follows.
* “queuewithlinkedlist” is the sub-package name under package “collections”.
* Create a class ”Main” with a main method as follows:







Assignment 07

* Do the Homework Assignment on Slide # 8 of *“5. Collection for Insertions - Queues”* presentation.

Assignment 08

\*\*\* This is a doSelect Test Problem.

##### **Mobile Shop**

 Coding

##### Description

Your task here is to implement ****Java**** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider ****default visibility**** of classes, data fields, and methods unless mentioned.

## Specifications

## 

## Task

****Class**** ****Mobile****

****-define the object of HashMap<String, ArrayList<String>>**** with variable name ****mobiles****.

* The ****String****defines the ****name of the company****and the ****Arraylist<String>****will have list of models.

****Implement the below methods for this class:****

****-String addMobile(String company, String model):****

* Write a code to add ****a company**** and its ****model****in ****mobiles****map as given below
* If the ****company**** does not exist in the map already, add the ****company**** and its ****model**** into the map. (Note: Add ****model**** into a new ****ArrayList<String>****and add this list into map as value)
* If the ****company**** already exist in the map, append the given ****model**** into the corresponding model list.
* Return "****model successfully added****" after performing the above operations

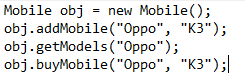
****-ArrayList<String> getModel(String company):****

* Write a code to get the Model list for the given company from Map ****mobiles****.
* Return ****null**** if the given company doesn't exist or doesn't have any model, else return the****List<String>**** of all the models.

****-String buyMobile(String company, String model):****

* Write a code to buy a mobile.
* Remove the mobile ****model**** from the list according to the ****compnay**** and ****model**** given. In case there are two same models then remove one and return the message "mobile sold successfully"
* Return a message "****item not available" i****f the ****company**** or corresponding ****model**** is not present in the Map

****Sample Input****



****Sample Output****



## NOTE:

* You can make suitable function calls and use ****RUN CODE**** button to check your ****main()**** method output.

##### Execution time limit

10 seconds

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class Mobile{

// Write your code here..

}

public class Source {

public static void main(String args[] ) throws Exception {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

}

}

Assignment 09

\*\*\* This is a doSelect Test Problem.

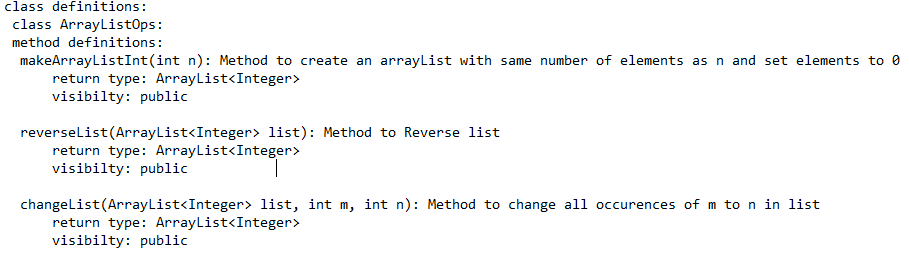
##### **List of Operations**

 Coding

##### Description

Your task here is to implement a ****Java**** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

****Specifications:****



****Task:****

Your task is to create a class ****ArrayListOps**** and implement the following:

****1.**** ****makeArrayListInt(int n):****Method to create an A****rrayList**** with number of elements as n and **set**elements to ****0****.

* If number of elements ****n**** is 4 , then the method should return a list containing ****[0,0,0,0]****

****2.**** ****reverseList(ArrayList<Integer> list):**** Method to **Reverse**list****

****3. changeList(ArrayList<Integer> list, int m, int n)****: Method to change all ****occurences**** of ****m**** to ****n**** in ****list****

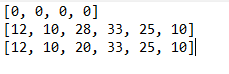
****Important:****

* To check your program, you can use the ****main()**** method (in Source class) given in the stub. You can make suitable function calls and use ****RUN CODE**** button to check your main() function output.

****Sample Input****



****Sample Output****



****NOTE:****

* The above ****Sample Output**** is only for demonstration purposes and will be obtained if you implement the ****main()**** method with all method calls accordingly.
* Upon implementation of ****main()**** method, you can use the ****RUN CODE**** button to pass input data in the method calls and arrive at the ****Sample Output****.

##### Execution time limit

10 seconds

import java.util.\*;

class ArrayListOps {

}

public class Source{

public static void main(String[] args) {

}

}

Assignment 10

\*\*\* This is a doSelect Test Problem.

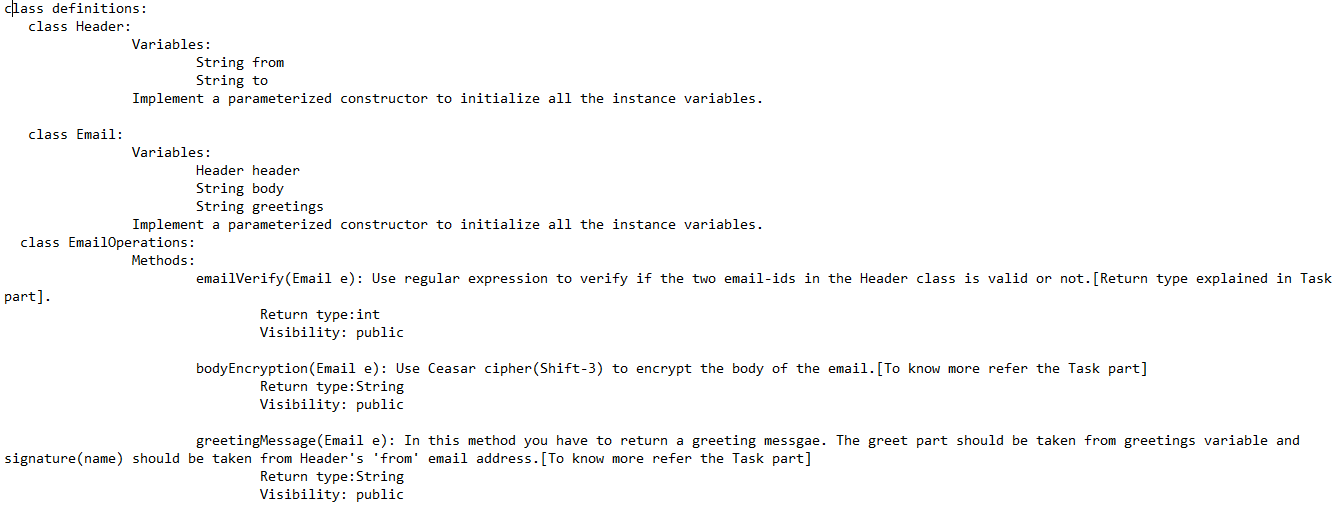
##### **Email Operation**

 Coding

##### Description

Your task here is to implement a Java code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications:



Class Variables:

* ****class Header:**** It contains two email id 'from' and 'to'. 'from' signifies the sender's email address and 'to' signifies receiver's email address.
* ****class Email:**** This class contains three parts: first Header header which has two email address from and to,the second body which contains the message to send and third greetings which contains greetings such as "Regards", "Thank you", etc.

To access a variable in Header class through Email object we use:



Example to access "from" address from the Email object e we use : e.header.from;

Tasks:

* Implement the two classes Email and Header class according to the specifications.
* Implement the three methods in the EmailOperations class:

1. emailVerify (Email e)
2. bodyEncryption (Email e)
3. greetingMessage (Email e)

Method Description:

1. emailVerify(Email e):

* In this method you have to use regex to check if the email-address to and from in Header class is valid or not. Validation is based on:
* Email address should start with alphabets(capital/small) or \_(underscore).
* Email address should have only one @ followed by alphabets.
* Email address should end with .(dot) followed by alphabets.
* e.g: amit@doselect.com, \_ami@doselect.in are valid addresses, but 1ami@dos.com, amit@doselect are invalid addresses.
* Return 2 if the both email addresses are valid return 1 if one is valid, and 0 if both are invalid.

2. bodyEncryption(Email e):

* In this method, you have to use Caesar cipher(shift of 3) to encrypt the body part of the Email return the encrypted string.
* Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. Here the number of shift is 3.
* e.g: str = "Hi There Hows you", after encryption becomes "Kl Wkhuh Krzv brx". H get converted to K that is a shift of 3 alphabets ahead.
* Letters which are capital should be capital and small should be small in Encrypted message. Take care of the spaces.

3. greetingMessage(Email e):

* In this method, you have to return a concatenated string which contains the greetings variable from Email class and Name of the person who is sending the mail(from variable in the Header class).
* The name part should not contain anything which is after @ in the email id.
* e.g: if greetings = "Regards" and from = "Amit@doselect.com" then you have to return the message "Regards Amit"

Important:

* To check your program you have to use the main() function(in Source class) given in the stub. You can make suitable function calls and use ****RUN CODE**** button to check your ****main()**** function output.

##### Execution time limit

10 seconds

REPORT AN ISSUE

class Email{

// Implement Email Class according to the specifiaction.

}

class Header{

// Implemet the Header Class according to the specifiaction.

}

class EmailOperations{

// Implemet the Three methods specified in the specified.

}

public class Source {

public static void main(String args[] ) throws Exception {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

// You can Implement your main() to check your Program.

Assignment 11

\*\*\* This is a doSelect Test Problem.

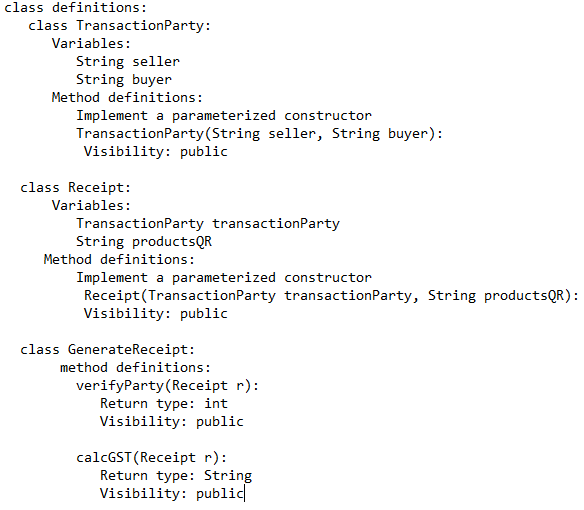
##### **Validating Users**

 Coding

##### Description

Your task here is to implement a****Java**** code based on the following specifications. Note that your code should match the specifications in a precise manner.  Consider****default visibility**** of classes, data fields and methods unless mentioned otherwise.

## ****Specifications****



****Tasks****

Class ****TransactionParty****

****-****define the ****String**** variable ****seller****

****-****define the ****String**** variable ****buyer****

-implement a parameterized ****constructor****

****class Receipt****

-declare the below instance variables and implement a parameterized constructor to initialize them.

1. ****TransactionParty**** ****transactionParty****

2. ****String**** ****productsQR****

* a string of the format: "****<Rate>,<Quantity>@<Rate>,<Quantity>@<Rate>,<Quantity>****"
* ****e.g:**** "250,10@100,3@50,7"
* only 3 products' ****Rate**** and ****Quantity**** present in the string

To access a variable in ****TransactionParty**** class through ****Receipt**** object we use:

****e.g:**** To access "****seller****" name from the ****Receipt**** object ****r**** we use:



class ****GenerateReceipt****

-Implement the below methods for this class:

****1.int verifyParty(Receipt r):****

In this method, you have to use regex to check if the names -****buyer****and ****seller****of ****TransactionParty**** object available in ****Receipt r****, are valid or not.Validate the names as per the below condition:

****Names:****

* should start and end with alphabets (uppercase/lowercase)
* can have white-spaces in between alphabets
* can have a single quote symbol OR a hyphen symbol in between alphabets
* ****e.g: Daniel D'Cruz, Giselle Dawn-Wright**** and ****Giselle Dawn****are valid names.

****Return:****

* 2 if the both names are valid
* 1 if only one name is valid
* 0 if both names are invalid.

****2. String calcGST(Receipt r):****

In this method, you have to use the ****productsQR**** variable of the ****Receipt**** ****r**** object to extract the ****Quantity**** and ****Rate**** of 3 products.

* The variable ****productsQR****will have a string value of the format ****<Rate>,<Quantity>@<Rate>,<Quantity>@<Rate>,<Quantity>****
* ****Rate**** is the price rate of the product
* ****Quantity**** is the number of units bought
* ****e.g****: ****productsQR**** = "****250,10@100,3@50,7****" has the ****Rate**** as 250, 100 and 50 while ****Quantity**** as 10, 3 and 7 respectively.

Use the ****GST\_Rate**** as ****12%.****

Calculate the value of GST using the formula:



The data type of GST should be ****int**** only. (Do NOT round-off the result)

Return the value of ****GST**** as a string value, using the ****toString()**** method.

****NOTE****

* You can make suitable function calls and use ****RUN CODE**** button to check your ****main()**** method output.

##### Execution time limit

10 seconds

import java.util.\*;

import java.lang.\*;

import java.util.regex.\*;

class TransactionParty {

}

class Receipt{

}

class GenerateReceipt{

}

class Source{

public static void main(String[] args){

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*