VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

SCHOOL OF INFORMATION TECHNOLOGY & ENGINEERING
FALL SEMESTER 2013-14
PROGRAMMING IN JAVA (LABORATORY) – ITA 418
CYCLE SHEET

SIMPLE PROGRAMS

1. Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters. Write a program that prompts the user to enter a weight in pounds and height in inches and display the BMI. Note that one pound is **0.45359237** kilograms and one inch is **0.0254** meters.

2. Solve the following system of linear equations

   $ax + by = e$

   $cx + dy = f$

   using Cramer's rule $x = (ed - bf)/(ad - bc)$ and $y = (af - ec)/(ad - bc)$. Write a program that prompts the user to enter **a**, **b**, **c**, **d**, **e**, and **f** and display the result. If ad - bc is **0**, report that "The equation has no solution".

3. The program randomly generates a number **0**, **1**, or **2** representing scissor, rock, and paper. The program prompts the user to enter a number **0**, **1**, or **2** and displays a message indicating whether the user or the computer wins, loses, or draws.

4. Write a program that simulates picking a card from a deck of **52** cards. Your program should display the rank (**Ace**, **2**, **3**, **4**, **5**, **6**, **7**, **8**,**9**, **10**, **Jack**, **Queen**, **King**) and suit (**Clubs**, **Diamonds**, **Hearts**, **Spades**) of the card.

5. Find the sum of the following series: $1/3 + 3/5 + 5/7 + 7/9 + \ldots + 97/99$

6. Write a program that reads integers, finds the largest of them, and counts its occurrences. Assume that the input ends with number **0**. Suppose that you entered **3 5 2 5 5 5 0**; the program finds that the largest is **5** and the occurrence count for **5** is **4**.

7. Write a program to find the mean and standard deviation for an array of 10 integers. The mean is simply the average of the numbers. The standard deviation is a statistic that tells how the various data are clustered around the mean in a set of data.

Mean(μ) = $\frac{\sum_{i=1}^{n} x}{n}$

Standard Deviation $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$

8. Write a program that prompts the user to enter the month and year and displays the number of days in the month. For example, if the user entered month **2** and year **2000**, the program should display that February 2000 has 29 days. If the user entered month **3** and year **2005**, the program should display that March 2005 has 31 days.

9. An *emirp* (prime spelled backward) is a nonpalindromic prime number whose reversal is also a prime. For example, **17** is a prime and **71** is a prime. So, **17** and **71** are emirps. Write a program that displays the first **100** emirps. Display **10** numbers per line

10. Develop a program to play lottery. The program randomly generates a lottery of a two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rule:
    - If the user input matches the lottery in exact order, the award is $10,000.
    - If all the digits in the user input match all the digits in the lottery, the award is $3,000.
    - If one digit in the user input matches a digit in the lottery, the award is $1,000.

STRINGS:

1. Write a program that prompts the user to enter a social security number in the format DDD-DD-DDDD, where D is a digit. The program displays **"Valid SSN"** for a correct social security number and **"Invalid SSN"** otherwise.

2. Some Websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rule is as follows:
    - A password must have at least eight characters.
    - A password consists of only letters and digits.
    - A password must contain at least two digits.

    Write a program that prompts the user to enter a password and displays **"Valid Password"** if the rule is followed or **"Invalid Password"** otherwise.

3. Write a method that finds the number of occurrences of a specified character in the string using the following header: **public static int** count(String str, **char** a). For example, **count("Welcome", 'e')** returns **2**. Write a test program that prompts the user to enter a string followed by a character and displays the number of occurrences of the character in the string.

4. Write a method that counts the occurrences of each digit in a string using the following header:

   public static int[] count(String s)

   The method counts how many times a digit appears in the string. The return value is an array of ten elements, each of which holds the count for a digit. For example, after executing **int[] counts = count("12203AB3")**, **counts[0]** is **1**, **counts[1]** is **1**, **counts[2]** is **2**, **counts[3]** is **2**. Write a test program that prompts the user to enter a string and displays the number of occurrences of each digit in the string.

5. The international standard letter/number mapping found on the telephone is shown below:

1	2	3
	ABC	DEF
4	5	6
GHI	JKL	MNO
7	8	9
PQRS	TUV	WXYZ
	0	

   Write a method that returns a number, given an uppercase letter, as follows:

   public static int getNumber(**char** uppercaseLetter).

   Write a test program that prompts the user to enter a phone number as a string. The input number may contain letters. The program translates a letter (upper- or lowercase) to a digit and leaves all other characters intact. For example : If the input is 1800flowers, the output is 18003569377

6. Write a method that parses a binary number as a string into a decimal integer. Invoke this in the main() method.The method header is as follows:

   public static int binaryToDecimal(String binaryString)

   For example, for the binary string 10001, the method should return 17. Note: Do not use the predefined method.

7. Write a method that parses a binary number into a hex number. The method header is as follows:

   public static String binaryToHex(String binaryValue)

8. Write a method that returns a sorted string using the following header:

   public static String sort(String s)

   For example, **sort("acb")** returns **abc**. Write a test program that prompts the user to enter a string and displays the sorted string.

9. Write a program that repeatedly prompts the user to enter a capital for a state. Upon receiving the user input, the program reports whether the answer is correct. Assume that **50** states and their capitals are stored in a two-dimensional array. The program prompts the user to answer all ten states' capitals and displays the total correct count.

10. Write a method that returns the common prefix of two strings. For example, the common prefix of **"distance"** and **"disinfection"** is **"dis"**. The header of the method is as follows:

    public static String prefix(String s1, String s2)

    If the two strings have no common prefix, the method returns an empty string. Write a **main** method that prompts the user to enter two strings and display their common prefix.

CLASSES AND OBJECTS

1. Design a class named **Rectangle** to represent a rectangle. The class contains:
   - Two **double** data fields named **width** and **height** that specify the width and height of the rectangle. The default values are **1** for both **width** and **height**.
   - A no-arg constructor that creates a default rectangle.
   - A constructor that creates a rectangle with the specified **width** and **height**.
   - A method named **getArea()** that returns the area of this rectangle.
   - A method named **getPerimeter()** that returns the perimeter.

   Implement the class. Write a test program that creates two **Rectangle** objects—one with width **4** and height **40** and the other with width **3.5** and height **35.9**. Display the width, height, area, and perimeter of each rectangle in this order.

2. Design a class named **Stock** that contains:
   - A string data field named **symbol** for the stock's symbol.
   - A string data field named **name** for the stock's name.
   - A **double** data field named **previousClosingPrice** that stores the stock price for the previous day.
   - A **double** data field named **currentPrice** that stores the stock price for the current time.
   - A constructor that creates a stock with specified symbol and name.
   - A method named **getChangePercent()** that returns the percentage changed from **previousClosingPrice** to **currentPrice**.

Implement the class. Write a test program that creates a **Stock** object with the stock symbol JAVA, the name Sun Microsystems Inc, and the previous closing price of **4.5**. Set a new current price to **4.35** and display the price-change percentage.

3. Design a class named **Account** that contains:
    - A private **int** data field named **id** for the account (default **0**).
    - A private **double** data field named **balance** for the account (default **0**).
    - A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0**). Assume all accounts have the same interest rate.
    - A private **Date** data field named **dateCreated** that stores the date when the account was created.
    - A no-arg constructor that creates a default account.
    - A constructor that creates an account with the specified id and initial balance.
    - The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
    - The accessor method for **dateCreated**.
    - A method named **getMonthlyInterestRate()** that returns the monthly interest rate.
    - A method named **withdraw** that withdraws a specified amount from the account.
    - A method named **deposit** that deposits a specified amount to the account.

    Implement the class. Write a test program that creates an **Account** object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the **withdraw** method to withdraw $2,500, use the **deposit** method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

4. Design a class named **Fan** to represent a fan. The class contains:
    - Three constants named **SLOW**, **MEDIUM**, and **FAST** with values **1**, **2**, and **3** to denote the fan speed.
    - A private **int** data field named **speed** that specifies the speed of the fan (default **SLOW**).
    - A private **boolean** data field named **on** that specifies whether the fan is on (default **false**).
    - A private **double** data field named **radius** that specifies the radius of the fan (default **5**).
    - A string data field named **color** that specifies the color of the fan (default **blue**).
    - A no-arg constructor that creates a default fan.
    - A method named **toString()** that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns fan color and radius along with the string “fan is off” in one combined string.

Implement the class. Write a test program that creates two **Fan** objects. Assign maximum speed, radius **10**, color **yellow**, and turn it on to the first object. Assign medium speed, radius **5**, color **blue**, and turn it off to the second object. Display the objects by invoking their **toString** method.

5. In a n-sided regular polygon all sides have the same length and all angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named **RegularPolygon** that contains:
    - A private **int** data field named **n** that defines the number of sides in the polygon with default value **3**.
    - A private **double** data field named **side** that stores the length of the side with default value **1**.
    - A private **double** data field named **x** that defines the x-coordinate of the center of the polygon with default value **0**.
    - A private **double** data field named **y** that defines the y-coordinate of the center of the polygon with default value **0**.
    - A no-arg constructor that creates a regular polygon with default values.
    - A constructor that creates a regular polygon with the specified number of sides and length of side, centered at (**0**, **0**).
    - A constructor that creates a regular polygon with the specified number of sides, length of side, and x-and y-coordinates.
    - The method **getPerimeter()** that returns the perimeter of the polygon.
    - The method **getArea()** that returns the area of the polygon. The formula for computing the area of a regular polygon is

$$Area = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}.$$

Implement the class. Write a test program that creates three **RegularPolygon** objects, created using the no-arg constructor, using **RegularPolygon(6, 4)**, and using **RegularPolygon(10, 4, 5.6, 7.8)**. For each object, display its perimeter and area.

6. Design a class named **QuadraticEquation** for a quadratic equation The class contains:
    - Private data fields **a**, **b**, and **c** that represents three coefficients.
    - A constructor for the arguments for **a**, **b**, and **c**.
    - Three **get** methods for **a**, **b**, and **c**.
    - A method named **getDiscriminant()** that returns the discriminant, which is $\mathbf{b^2 - 4ac}$.
    - The methods named **getRoot1()** and **getRoot2()** for returning two roots of the equation

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ and } r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

These methods are useful only if the discriminant is nonnegative. Let these methods return **0** if the discriminant is negative. Implement the class. Write a test program that prompts the user to enter values for a, b, and c and displays the result based on the discriminant. If the discriminant is positive, display the two roots. If the discriminant is **0**, display the one root. Otherwise, display "The equation has no roots."

7.

(*Algebra:* 2×2 *linear equations*) Design a class named `LinearEquation` for a 2×2 system of linear equations:

$$ax + by = e \qquad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$
$$cx + dy = f$$

The class contains:

- Private data fields `a`, `b`, `c`, `d`, `e`, and `f`.
- A constructor with the arguments for `a`, `b`, `c`, `d`, `e`, and `f`.
- Six `get` methods for `a`, `b`, `c`, `d`, `e`, and `f`.
- A method named `isSolvable()` that returns true if $ad - bc$ is not `0`.
- Methods `getX()` and `getY()` that return the solution for the equation.

Implement the class. Write a test program that prompts the user to enter **a**, **b**, **c**, **d**, **e**, and **f** and displays the result. If **ad-bc** is **0**, report that "The equation has no solution."

8. In a school, students of all classes from std I to X appear for the MathPremierLeague examination. Define a class MPL which stores the details of the marks scored by each class. It should contain the following data members:
    - Standard, an integer
    - No. of students, an integer
    - An integer array to store the score of each student in MPL exam
    - First mark, an integer
    - Create an array of 10 objects for this class.
    - Provide only a parameterized constructor which receives the values for all its data members from the main() method.
    - Define a method findBestClass() to display the standard which has secured the highest first mark.

Implement the class. Write a test program to get the no. of students from each class and their respective MPL marks, and find the standard which has secured the highest first mark

9. Define a class called student with the following attributes:

    - Roll number
    - Name
    - Date of birth
    - Weight
    - Height
    - Mark

    Write a suitable parameterized constructor to initialize the object and a method to display the details of a student object. In the main method create some student objects and display the roll numbers and the name of the students who are 19 years old or more with weight above 90.5 kg but height less than 175.0 cm.

10. Define a class vector (not the predefined class in Java) with the following instance variables

    - x
    - y
    - Define 2 constructors : To assign 1 for all the components of the vector and the other to assign user defined values for x and y components

    Create two vector objects ‘v1’ and ‘v2’ to work in two dimensions. i.e. x and y values alone should be considered. Create two more objects ‘v3’ and ‘v4’ having all components Define a method named dotProduct() to compute the dot product of two 2D vectors as follows: If vector U=ai+bj and V=ci+dj then U.V=ac+bd.

THREADS

1. Write a program that creates three tasks and three threads to run them:
    - The first task prints the letter *a* 100 times.
    - The second task prints the letter *b* 100 times.
    - The third task prints the integers 1 through 100.

2. Write a program that creates three tasks and three threads to run them:
    - count the number of characters (excluding control characters '\r' and '\n') in a file
    - count the number of words in a file
    - count the number of lines in a file

3. Write a program that creates three tasks and three threads to run them:
    - Read and display the elements (integers) of an array of size 10.
    - Display the total of the values in the array.
    - Display the average of the values in the array.

4. Write a program that creates two tasks and two threads to run them:
    - Read and display the elements (integers) of an array of size 10 from index 0 to 9.
    - Read and display the elements (integers) of an array of size 10 from index 9 to 0

5. Write a program that creates two tasks and two threads to run them:
    - Sort an array of integers using bubble sort.
    - Sort an array of integers using selection sort.

6. Write a program that creates two tasks and two threads to run them:
    - Search an array of integers for an integer using linear search.
    - Search an array of integers for an integer using binary search.

7. Write a program that creates three tasks and three threads to run them:
    - Display numbers from 1 – 10.
    - Display squares of numbers from 1 – 10.
    - Display cubes of numbers from 1 – 10.

8. Write a program that creates two tasks and two threads to run them:
    - Convert decimal to binary.
    - Convert decimal to hex.

9. Write a program that creates two tasks and two threads to run them:
    - Read file from beginning to end.
    - Read file from end to beginning.

10. Write a program that creates two tasks and two threads to run them:
    - Calculate area of rectangle.
    - Perimeter of rectangle.

FILE I/O

1. Write a program that will count the number of characters (excluding control characters '\r' and '\n'), words, and lines, in a file.

2. Write a program that reads the scores from the file and displays their total and average. Scores are separated by blanks.

3. Write a program to create a file and write 100 integers created randomly into the file. Integers are separated by spaces in the file. Read the data back from the file and display the sorted data using selection sort.

4. Write a program to create a file and write 100 integers created randomly into the file. Integers are separated by spaces in the file. Read the data back from the file and display the sorted data using bubble sort.

5. Write a program that replaces text in a source file and saves the change into a new file.

6. Write a program that removes all the occurrences of a specified string from a text file.

7. Write a program to check if 2 text files have the same contents.

8. Write a program to merge two text files.

9. Write a program to swap contents between 2 text files.

10. Write a program to reverse the contents of a text file.

EXCEPTION HANDLING, PACKAGES, JDBC, RMI, APPLETS, SWINGS, JSP, SERVLETS

For questions 1 – 10 in **CLASSES AND OBJECTS** section, implement the above concepts as described below:

Consider Q.1 for example:

- For applet, swings, JSP, servlets, design the GUI with 2 labels (height and width), 4 textbox (height, width, area, perimeter) and 2 buttons (calculate area, calculate perimeter).
- For exception handling, package, JDBC, RMI, get the height and width from console, as input.
- Implement the business logic for calculating area and perimeter using respective technology.
- For exception handling, throw exception if height or width is not in correct format.
- For JDBC, store the result inDB.
- For exception handling, package, JDBC, RMI, display output in console.
- For applet, swings, JSP, servlets, display the output in GUI.

JAVA NETWORKING

1. Write a Java program to print the IP address of a given host.
2. Write a Java program to print the name of a given host.
3. Write a Java program to print the class of IP address.
4. Write a Java program to identify IP address of both client and server.
5. Write a Java program to implement Date & Time Server
6. Write a Java program to provide a communication between client and server where they can send only a single message between each other.
7. Write a Java program to implement TCP chat application.
8. Write a Java program to implement UDP chat application.
9. Write a Java program to implement Echo UDP Server
10. Write a Java program to handle the file transfer between the client and the server.

GRAPHICS PROGRAMMING

1. Write a Java program to draw a container.
2. Write a Java program to draw hills.
3. Write a Java program to draw a chess board.
4. Write a Java program to draw a smiley.
5. Write a Java program to draw an Olympic circle.
6. Write a Java program to draw a doll.
7. Write a Java program to draw a house.
8. Write a Java program to draw a car.
9. Write a Java program to draw a star.
10. Write a Java program to draw a national flag

INHERITANCE

1. Create a class **Student** with the following attributes**:** Name, Reg_no and Marks[6]. Define a method getInfo() to read the above details for all the students of a class. Derive another class **Exam** with a single attribute – current CGPA. Define a method CalculateCGPA( ) to first find the GPA of a student in that semester and hence his new CGPA.

2. In the weekends, students learn extra-curricular activities such as Guitar, Keyboard, Veena and Carnatic music. Create a class named **Teacher** to include the following fields: Name, Course_taught, timing and number_of_students who have joined that course. Create another class **Musician** with the following instance variables**:** KeyBoard_fees, Guitar_fees, Veena_fees and CarnaticMusic_fees and an array containing the count of the students who have joined in each course. Assuming a student can join for more than one music course, define a method CalculateFee() to read the number of courses learnt by each student and the fees for those courses and thus calculate the total fees to be paid by the student. Another method should also be defined to find the total monthly income of the music teacher.

3. Class **Person** should be created with the fields – Name, DOB and address. It also should include a method to read these details and another method to calculate his age at present. Another class **Employee** should be derived from this class with the following additional attributes: designation, school_name, salary, Date_of_join and currentsem_course. Define a method to calculate his years of experience and finally display all his details.

4. Design a Class **Salaried employee** with the fields: Employee number, name, designation, basic pay, HRA, DA and gross pay. Include a method to read the name of the employee, number, designation and basic pay. Also have another method to calculate HRA and DA based on the basic pay and thus evaluate his gross pay. Class **Salaried commissioned employee** should inherit salaried employee class. A method should be declared to read if it is a flat commission or ramped commission. Declare another method to calculate the commission for the employee. For flat commission, commission percentage is 7.5 percent, on any sale the representative makes. For ramped commission, the percentage of commission should be as follows: 10 percent commission on the first $50,000 worth of goods or services, a 15 percent commission on the next $50,000 and a 20 percent commission on anything above $100,000. Define a method to find the actual salary by taking 30 percent income from base salary and 70 percent from commission. The commission should be calculated from the method declared in the interface.

5. Design a Class **Scanner** with the instance variables – colordepth, densityrange and resolution. Include a method checkDensityRange() to read the type of the scanner (consumer level flat bed scanner, high end flat bed scanner or drum scanner) and check if the density range entered by the user is valid or not, based on the following conditions: Consumer level flatbed scanners should have a Drange in the 2.5-3.0 range. High end flatbed scanners can reach a Drange of 3.7. Drum scanners should have a Drange of 3.6 - 4.5. Extend the Scanner class to create a new class **Copier** which has its own attributes such as responsetime, papercapacity and type (color/ B&w / hybrid) and with the following methods: getPrinterType() to read if it is an ink jet or laser printer and getSpeedAndResolution() to read the speed and resolution (dots per inch) of the printer. Also define a method to suggest a suitable printer, copier and scanner for a business concern based on the requirements.

6. Design a Class **Student** with the attributes mentioned below**:** Name, Reg_no, Programme and Semester. Also have a method getDetails() defined to read these details. Create another class **Exam** derived from Student with the following instance variables: noOfCourses (to read the count of the courses registered this semester) and isRegisteredForSports (a boolean variable) and 'Credits' (an integer) and two methods getSportName() and getGrade(). Also include a method evaluateGPA() to find the GPA obtained by the student in that semester. This method should check the Boolean variable and if it is true should get the grade of the student in the sports and include it while calculating the GPA.

7. The **Account** class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn. Implement the classes. Write a test program that creates objects of **Account**, **SavingsAccount**, and **CheckingAccount** and invokes their **toString()** methods.

INTERFACES

1. Create a class **Student** with the following attributes**:** Name, Reg_no and Marks[6]. Define a method getInfo() to read the above details for all the students of a class. Develop an Interface **Exam** with the method CalculateGPA( ) to find the GPA of a student in that semester. Create another class **Result** which extends Student class and implements the Exam interface. It should also contain the method Display( ) to print the student details.

2. In the weekends, students learn extra-curricular activities such as Guitar, Keyboard, Veena and Carnatic music. Create a class named **Teacher** to include the following fields: Name, Course_taught, timing and number_of_students who have joined that course. Develop an Interface **Musician** with the following static and final variables**:** KeyBoard_fees=500, Guitar_fees=300, Veena_fees=400 and CarnaticMusic_fees=250. Declare a method CalculateFee() to calculate the total fees to be paid by the student. Create another class **Musical teacher** which implements Musician and extends Teacher. This class should read the number of courses learnt by each student and thus define the method CalculateFee() to find the total fees. A method should also be defined to find the monthly income of each teacher.

3. Class **Person** should be created with the fields – Name, DOB and address. It also should include a method to read these details and another method to calculate his age at present. Define an Interface **Employee** with the final member- InstitutionName with value "VIT". Declare a method getSchoolName() to read the name of the school. Another Class **Teacher** should extend Person class and implement Employee interface. It should also contain additional attributes such as designation, salary, Date_of_join and currentsem_course. Define a method to calculate his years of experience and finally display all his details.
4. Class **Salaried employee** with the fields: Employee number, name, designation, basic pay, HRA, DA and gross pay. Include a method to read the name of the employee, number, designation and basic pay. Also have another method to calculate HRA and DA based on the basic pay and thus evaluate his gross pay. Define an Interface **Commissioned employee.** A method should be declared to read if it is a flat commission or ramped commission. Declare another method to calculate the commission for the employee. For flat commission, commission percentage is 7.5 percent, on any sale the representative makes. For ramped commission, the percentage of commission should be as follows: 10 percent commission on the first $50,000 worth of goods or services, a 15 percent commission on the next $50,000 and a 20 percent commission on anything above $100,000. Class **Salaried commissioned employee** should inherit salaried employee class and implement commissioned employee interface. Define a method to find the actual salary by taking 30 percent income from base salary and 70 percent from commission. The commission should be calculated from the method declared in the interface.

5. Class **Scanner** with the instance variables – colordepth, densityrange and resolution. Include a method checkDensityRange() to read the type of the scanner (consumer level flat bed scanner, high end flat bed scanner or drum scanner) and check if the density range entered by the user is valid or not, based on the following conditions: Consumer level flatbed scanners should have a Drange in the 2.5-3.0 range. High end flatbed scanners can reach a Drange of 3.7. Drum scanners should have a Drange of 3.6 - 4.5. An Interface **Printer** should be developed with the following methods: getPrinterType() to read if it is an ink jet or laser printer and getSpeedAndResolution() to read the speed and resolution (dots per inch) of the printer. Extend the Scanner class and implement the Printer interface to create a new class **Copier** which has its own attributes such as responsetime, papercapacity and type (color/ B&w / hybrid). Also define a method to suggest a suitable printer, copier and scanner for a business concern based on the requirements.

6. Design a Class **Student** with the attributes mentioned below**:** Name, Reg_no, Programme and Semester. Also have a method getDetails() defined to read these details. Create an Interface **Sports** with a final member 'Credits' with value 2 and two methods getSportName() and getGrade(). Create another class **Exam** derived from Student and implemented from Sports with the following instance variables: noOfCourses (to read the count of the courses registered this semester) and isRegisteredForSports (a boolean variable). Include a method evaluateGPA() to find the GPA obtained by the student in that semester. This method should check the Boolean variable and if it is true should get the grade of the student in the sports and include it while calculating the GPA.

7. Create a class named **Current**A**ccount** to model a bank account. It must have the following properties: account number, balance, and date created, and methods to deposit and withdraw funds. Create an interface named **SavingsAccount** with a static final member rateofinterest =0.04 and declare a method to get the balance in this account. Extend CurrentAccount class and implement SavingsAccount interface to create a new class **PremiumAccount.** This class should define a method to find the total balance (balance in current account + balance in savings account) after 5 years.