CODEX, SIT PUNE
JAVA ASSIGNMENT

1. A super class Bank has been defined to store the details of a customer. Define a sub-class
   Account that enables transactions for the customer with the bank. The details of both the
   classes are given below:
   Class name : Bank
   Data member/instance variable:
   name : stores the name of the customer
   accno : stores the account number
   p : stores the principal amount in decimals
   Member functions/methods:
   Bank(...) : parameterized constructor to assign values to the instance variables
   void display( ) : displays the details of the customer
   Class name: Account
   Data member/instance variable:
   amt : stores the transaction amount in decimals
   Member functions/methods:
   Account(...) : parameterized constructor to assign values to the instance variables of both the classes
   void deposit( ) : accepts the amount and updates the principal as p=p + amt
   void withdraw( ) : accepts the amount and updates the principal as p=p-amt
   If the withdrawal amount is more than the principal amount, then display the message
   "INSUFFICIENT BALANCE". If the principal amount after withdrawal is less than 500, then a penalty is imposed by using the formula p=p-(500-p)/10
   void display( ) : displays the details of the customer
   Assume that the super class Bank has been defined. Using the concept of Inheritance, specify the class Account giving details of the constructor(...), void deposit( ),
   void withdraw( ) and void display( ). The super class and the main function need not be written.

2. A superclass Number is defined to calculate the factorial of a number. Define a subclass Series to find the sum of the series S = 1! + 2! + 3! + 4! + ………. + n! [5]
   The details of the members of both classes are given below:
   Class name: Number
   Data member/instance variable:
   n: to store an integer number
   Member functions/methods:
   Number(int nn): parameterized constructor to initialize the data member n=nn
   int factorial(int a): returns the factorial of a number
   (factorial of n = 1 × 2 × 3 × …… × n)
   void display()

Class name: Series
Data member/instance variable:
sum: to store the sum of the series
Member functions/methods:
Series(…) : parameterized constructor to initialize the data members of both the classes
void calsum(): calculates the sum of the given series
void display(): displays the data members of both the classes
Assume that the superclass Number has been defined. Using the concept of inheritance, specify the class Series giving the details of the constructor(…), void calsum() and void display().
The superclass, main function and algorithm need NOT be written.

3. A superclass Product has been defined to store the details of a product sold by a wholesaler to a retailer. Define a subclass Sales to compute the total amount paid by the retailer with or without fine along with service tax. [5]
Some of the members of both classes are given below:
Class name: Product
Data members/instance variables:
name: stores the name of the product
code: integer to store the product code
amount: stores the total sale amount of the product (in decimals)
Member functions/methods:
Product (String n, int c, double p): parameterized constructor to assign data members: name = n, code = c and amount = p
void show(): displays the details of the data members
Class name: Sales
Data members/instance variables:
day: stores number of days taken to pay the sale amount
tax: to store the sen ice tax (in decimals)
totamt: to store the total amount (in decimals)
Member functions/methods:
Sales(….): parameterized constructor to assign values to data members of both the classes
void compute(): calculates the service tax @ 12.4% of the actual sale amount calculates the fine @ 2.5% of the actual sale amount only if the amount paid by the retailer to the wholesaler exceeds 30 days calculates the total amount paid by the retailer as (actual sale amount + service tax + fine)
void show (): displays the data members of the superclass and the total amount
Assume that the superclass Product has been defined. Using the concept of inheritance, specify the class Sales giving the details of the constructor (…), void compute()) and void show(). The superclass, main function and algorithm need NOT be written.

4. A superclass Worker has been defined to store the details of a worker. Define a subclass Wages to compute the monthly wages for the worker. The

details/specifications of both the classes are given below:
Class name: Worker
Data Members/instance variables:
Name: to store the name of the worker
Basic: to store the basic pay in decimals
Member functions:
Worker (…): Parameterised constructor to assign values to the instance variables
void display (): display the worker’s details
Class name: Wages
Data Members/instance variables:
hrs: stores the hours worked
rate: stores rate per hour
wage: stores the overall wage of the worker
Member functions:
Wages (…): Parameterised constructor to assign values to the instance variables of both the classes
double overtime (): Calculates and returns the overtime amount as (hours*rate)
void display (): Calculates the wage using the formula wage = overtime amount + Basic pay and displays it along with the other details
Specify the class Worker giving details of the constructor () and void display ( ).
Using the concept of inheritance, specify the class Wages giving details of constructor ( ), double-overtime () and void display (). The main () function need not be written.

5. A superclass Stock has been defined to store the details of the stock of a retail store. Define a subclass Purchase to store the details of the items purchased with the new rate and updates the stock. Some of the members of the classes are given below: [10]
Class name: Stock
Data members/instance variables:
item: to store the name of the item
qt: to store the quantity of an item in stock
rate: to store the unit price of an item
amt: to store the net value of the item in stock
Member functions:
Stock (…): parameterized constructor to assign values to the data members
void display(): to display the stock details
Class name: Purchase
Data members/instance variables:
pqty: to store the purchased quantity
prate: to store the unit price of the purchased item
Member functions/ methods:
Purchase(…): parameterized constructor to assign values to the data members of both classes
void update (): to update stock by adding the previous quantity by the purchased quantity and replace the rate of the item if there is a difference in the purchase rate. Also, update the current stock value as (quantity * unit price)

void display(): to display the stock details before and after updation.
Specify the class Stock, giving details of the constructor() and void display(). Using the concept of inheritance, specify the class Purchase, giving details of the constructor(), void update() and void display().
The main function and algorithm need not be written.