Core Java Exercises

General instructions:

1. *Follow coding standards, naming conventions for all the exercises.*
2. *Use comments wherever required.*
3. *Create default constructor, overloaded constructors, setter's & getter's, hash code & Boolean equals and toString methods for all classes.*
4. *Entity classes should implement Serializable interface.*
5. *Usage of input / output statements in classes are discouraged (except Main class)*
6. *Use Loggers in all exception handling statements in catch block (Log error messages to a .log file)*

1. Create an Entity class '**Trainee**', with traineeId, traineeName, contactNo, email, gender, age as fields.

   Create a class 'Batch', with batchCode, startdate, enddate and Trainee[] as fields. Create the following overloaded methods in the 'Batch' class

   public Trainee getTrainee(int traineeId) thows TraineeNotFoundException
   public Trainee[] getTrainees(String gender)

2. Follow the given instructions and create an application using Java.
   (i) Create an entity class named Project with member variables as projectId, projectName, projectHead, noOfResources.
   (ii) Create an object for the Project class and through setters assign the values for all the member variables.
   (iii) Print the corresponding object.

3. Prepare a **StringServiceProvider** class which has the following methods
   (a) To reverse a given string
   (b) To do linear search in a given string
   (c) To do search and replace operation in a given string
   Note: code the requirement with 2 possibilities (with and without static methods)

4. Follow the given instructions and create an application using Java.
   (i) Create a '**BankAccount**' class with 3 data members, accountNo[use String], accountName and balance.
   (ii) Overload the BankAccount constructor to accept only accountNo and accountName variables.
   (iii) Initialize the balance with 1000.

5. Follow the given instructions and create an application using Java.

   (i) Create a Bank Class having an array of BankAccount as a data member, populate the array through setters.

   (ii) Introduce a static variable called lastAssignedNo(integer). This should be initialized to 0 in the beginning. While creating new bank accounts, the accountNo

variable should not be supplied in the constructor parameter list. Instead it has to be computed as (lastAssignedNo + 1). Also modify the lastAssignedNo after creating a bank account.

(iii) Create an **IBankServiceProvider** interface and declare the following methods:

(a) **BankAccount checkAccount(String accountNo)**
This checks whether the given account number is available in the array or not. If exists, it should return the object of BankAccount class, else return null. *Reuse this method in all the other methods given below*.

(b) **double getBalance(BankAccount account)**
This will return the balance in an account for the given account

(c) **boolean depositMoney(BankAccount account, double amount)**
This deposits the given amount into the given account number after verifying whether the given account is present in the array or not.

(d) **boolean withdrawMoney(BankAccount account, double amount)**
This will withdraw the given amount from the given account after verifying the existence of account as well as balance.

(e) **boolean transferMoney(BankAccount fromAccount, BankAccount toAccount amount)**
This transfers the money from one account to another account after verifying both the accounts are existing or not as well as balance of the 'fromAccount'.

(iv) Bank class should implement **IBankServiceProvider** interface and override the methods of the interface.

6. Create user defined exceptions, **InsufficientFundException** and **InvalidAmountException** classes.

   Throw these exceptions from the methods in Bank class. Make necessary changes to accommodate these exception in the source code. Handle all these exceptions from the main program.

7. Using internationalization convert the date & time into Locale specific format.

8. Use resource bundle to acknowledge a user who did online shopping

   a) Good Morning <username>
   b) You ordered <Quantity requested > quantity of <product name> at <time stamp >
   c) The products you ordered will be delivered to <address>

9. Prepare currency convertor using internationalization

10. Validate the input with regular expressions.

    a. Email-ID
    b. Password
    c. Credit-card Number

11. Create a thread that does counter incrementation in its run() method, create three instances with three different priorities and see the effect of it in their outputs.

12. Demonstrate Producer-Consumer solution using threads.
    The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

    The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.

13. Design a class **HTCGlobalServices** which has the following attributes:
        Company ID
        Address
        EmployeeMap < id, Employee>

    → Employee is an Entity class
    → Map<Key, Value> is EmployeeMap<empID, Employee>
    → **HTCglobalservices** class should implement **ICompanyserviceprovider** to facilitate basic CRUD operations on Employee Class.
    (Try the same exercise with employee details as list and observe the difference in the code)

14. Understand the below given requirements and design the system with the required Entity classes that has data members, properties and member functions.

    Requirements:
    The proposed system will follow the general operations of the library like, maintaining different kinds of books categorized as fiction, novel and general by giving a unique book id and recording the book name, author name, publisher name and the number of copies of that book.

    Maintain the members who are allowed to take books from the library by giving a unique member id to each and every member and recording the member name, address and telephone number. Each member is given 3 cards, where one book can be taken on each card, and the user has to return the book within 7 days after the issue of the book, otherwise a fine of Rs. 2/- will be charged per day.

    This system should generate different kind of reports like, displaying the total no. of members, displaying available books of different categories, books due on the current date, and books issued to members.

    System has to take care of the following details.
        ➢ Maintaining Member details
        ➢ Maintaining Books details

➢ Transactions (Issue And Return of Book)
➢ Reports

15. Create a class **Stores** with an ArrayList<Product> as its only field, initialize the list in the constructor of stores. Create a method in the Stores class that returns the sorted list.

16. File '**issues.txt**' that contains comma separated issue records as follows
    issueId
    issueGeneratedDate
    issueCategory
    issueDescription
    Read each line of issue records and load into an ArrayList<Issue> in a Java program.

17. Create a class Employee with following fields
    employeeID, employeeName, password, salary, deptNO
    Create a unique collection of employees and count them based on the department number and list the result. Serialize the above collection of employees into a file. While serializing rotate the password by a number, while deserializing rotate it in reverse, using the same number.

18. Write a class **Product**, with following data members and member functions for accomplishing the given requirements.
    Fields:
    productId, productName, price, quantityOnHand, reorderLevel, reorderQty

    Assume reorderLevel of the product is 10 qty's and reorderQuantity is 50 quantities.

    Write a class **Stores** which has a List of Products as its only field. Populate the list in the constructor of Stores. Add the following methods to this class.

double sellItem(int productCode, int qtyRequired) throws ProductNotFoundException:
It should sell the required qty and show the total amount.
This method also checks whether reorderLevel is reached. If yes, raise a purchase order.
[Just print a message on the screen saying that 'purchase order is made'].

void updateStock(int productCode, int arrivedQty)
throws ProductNotFoundException;

19. Create a Comparator<Product> object to sort the products in stores class with respect to price. Display the sorted products.

20. There is a class called **Pet** which is not a Serializable class.
    There is another class called **Person** which is Serializable and has Pet as a field.
    Write a manual serialization program to serialize the Person object

21. A software company would like to develop a shopping application which supports the following functionality:

    A buyer can buy Products. To buy a product, the buyer specifies what product and how much quantity of it he/she needs (for e.g. Wheat Flour - 2 packets). He/She then adds this requirement to his Cart. The application should compute the total cost for the

products purchased by the customer.

Design the necessary classes, user defined exceptions to accomplish the requirement.

Don't suppress any exceptions inside the business layer. Chain the exceptions, if raised and handle them in the appropriate place.

22. Write a reflection based program to count the test methods (method with @Test annotation) available in a class.
23. Write a program to accept a directory name for reading all the class files available in the directory and count number of non parametrized public functions which has started with the word **test**.

24. Assume the following string contains department number and all its employee names in the below given format, split it and store it in the appropriate collection.
    {
        10:[“Surendran”, “Ankith”, “Bala”];
        20:[“Judini”, “Rajesh”, “Naveen”, “Naveen”];
        30:[“Koustav”, “Vignesh”];
        40:[“Rizwan”, “HimaSree”, “Mahesh”];
    }

25. Design a class Bike with following fields and member functions for accomplishing the given requirements.
    Fields: bikeRegNo, bikeName, maxSpeed, currentSpeed, currentGear
    Member Functions:
    applyBreak() :It should reduce the currentSpeed to zero and currentGear to 1
    accelerate(int speed): It should increase or decrease the speed based on the parameter.
    (For example if parameter speed is positive number, it should increase the speed else it should decrease the speed by parameter value).

    Along with changing the speed it should change the currentGear property also, based on the speed range.
        If currentSpeed is 0 to 10 then gear is 1.
        if currentSpeed is 11 to 20 then gear is 2.
        if currentSpeed is 21 to 40 then gear is 3.
        if currentSpeed is greater than 40 then gear is 4.

26. Write a class ParkedBike which is a subclass of Bike (A class written above) with following fields

        parkedDate
        deParkedDate

    Write a Class ParkingStation which has a collection of ParkedBike and following member functions
    parkBike: it should add a bike to the collection
    deParkBike: it should remove a bike it should return parking charges (Assume 5rs/day)

27. Design a class HTCEmployeeServices which implements IDBservices to facilitate basic CRUD operation on an employee table

    Prepare three user defined exception class

    a) "**EmployeeNotAvailableexception**", use it in Read, Update and Delete operations.

    b) "**DuplicateEmployeeException**", use it in Create operation.

    c) "**InvalidSalaryException**", use it in create operation, if the salary of an employee is negative. Log the exception messages and other acknowledgment messages to a file "HTCLOG.txt"

    Note:

    1. Do not hard code the connection string in the program
    2. Update operation should update salary of a give employee
    3. Use Callable statement to perform CRUD operations

28. There is a table Users [userName varchar(20) primary key,
    password varchar(15) not null,
    useraddress varchar(50),
    email varchar(20) not null].

    Write a Stored Procedure/ Function to insert a row in the table. Use Callable statement to invoke the procedure/function to insert records in the table. Provide all necessary validations.

29. Create a UserDAO with the following two methods

    public boolean validateUser(String username, String password) ;

This method validate the user with the given username & password and return true/false.

    public boolean registerUser(User user) ;

This method register/insert a new user in the database.

(Use Callable statement to perform CRUD operations)

30. Simulate deep copy and shallow copy with an entity class