1) Write a program to create a class named **Machine** having:
**private** attributes: company_name, date_of_manufacture
Use parameterized constructor to initialize these attributes. Make setter & getter methods for these attributes.

Now create a sub-class of **Machine** which is named **Vehicle** and having:
**private** attributes: vehicle_type, mileage, price
Use parameterized constructors to initialize these attributes and make setter & getter methods for these attributes also.

Make **Welcome** class consisting of main() method where you have to create at least 5 vehicle-objects and then serialize all these objects in a file named "records.doc". Now make another method named display() in the same class which will be used read all the objects from the file and display the details of only those vehicle-objects whose manufacturing date is greater than year: 2015.

**Note: Don't make any separate instance variable other than the ones specified in above scenario. Also, date_of_manufacture must be taken as a LocalDate object instead of String object. Also, make use proper exception handling by using inbuilt exception-classes and also make your own custom exception class to deal with the exceptions such as user entering a negative no.**

2) Write a program to create following scenario:
A class **Person** having:
**private** field: name → to store name of person
**abstract** method: void display() → which will be used to display details of person
Use parameterized constructor to initialize these attributes. Make setter & getter methods for the attribute.

Make a class **Employee** which inherits **Person** class and having:
**private** field: id → to store id of employee
date_of_join → to store joining date of employee
Use parameterized constructor to initialize these attributes. Make setter & getter methods for the attributes also.

Now make a class **Test** having main() method in which you have to create at least 6 Employee objects and store them in ArrayList. Make another method named serializeObjects() in the same class which will read all the employee-objects from the ArrayList and serialize all those employee-records (in file "records.txt") whose date of joining is before the year 2015.
Now make another method named deserializeObjects() in the same class which will read and display all the records from the file.

**Note: Don't make any separate instance variable other than the ones specified in above scenario. Also date_of_join variable is to be taken as LocalDate object rather than String object. Also, make use proper exception handling by using inbuilt exception-classes and also make your own custom exception class to deal with the exceptions such as user entering a negative no.**

3) Write a program to implement following scenario:
Create an interface **Colors** having:
**default** method: void showColor() → which prints color of an object

Create a class **Shape** having:
**abstract** method: double calculateArea() → will be used to calculate and return area

Create a class **Circle** which inherits **Shape** and **Colors** and has:
**private** field: radius → to store radius of circle
Use parameterized constructor to initialize the attribute. Make setter & getter methods for the attribute.

Create a class **Test** having main() method where you have to create at least 4 objects of **Circle** class and store them in Arraylist.
Now make another method named serializeObjects() which will read the objects from the ArrayList and serialize them in a file named "data.txt".
Now make another method named displayData() in the same class which will read the objects from the file and display the details of only those objects whose area is > 50.

**Note: Don't make any separate instance variable other than the ones specified in above scenario. Also, make use proper exception handling by using inbuilt exception-classes and also make your own custom exception class to deal with the exceptions such as user entering a negative no.**

4) Write a program to create following scenario:
**abstract** class **VehicleType** having:
**private** field: vehicle_type → (Eg: scooter, car, bus)
Use parameterized constructor to initialize the attribute. Make setter & getter methods for the attribute.

An interface **Color** having:
**default** method: getColor() → which returns color (default: "white") of vehicle

An interface **MfgDate** having:
**abstract** method: getMfgDate() → will be used to return manufacturing date of vehicle

Now create a class **Vehicle** which inherits **MfgDate, Color** & **VehicleType** and having:
**private** field: mfg_date → to store the manufacturing date of vehicle
Use parameterized constructor to initialize the attribute. Make setter & getter methods for the attribute.

Create **Test** class having main() method in which you have to make at least 5 objects of the **Vehicle** class and store them in ArrayList. Create another method named serializeVehicles()

which will store all objects in a file named “Records.txt”. Now make another method named deserializeVehicles() which will read and display all those objects from the file whose manufacturing-date is > 2018.

**Note: Don’t make any separate instance variable other than the ones specified in above scenario. Also mfg_date variable is to be taken as LocalDate object rather than String object. Also, make use proper exception handling by using inbuilt exception-classes and also make your own custom exception class to deal with the exceptions such as user entering a negative no.**

5) Write a program to read the contents of a file whose location is taken using command-line argument. Verify whether path exists or not. If path given is incorrect, then it should keep on asking the correct path from the user until user enter the correct path of a file/directory.

   Now, if the path entered is of the directory, then it should display the names of all files & directories in that directory. Now ask the user to enter the name of a file and verify if the file mentioned exists this directory or not. If it exists, then perform the following task:

   a) Print the following information about the file: size, readable/writable.
   b) Ask the user to enter a message and save this message in the file such that previous contents of the file are not erased.
   c) Now read the contents of this file display them on the console screen.

   **Note: Make use of Character-stream class to read/write contents. Don’t make any separate instance variable other than the ones specified in above scenario. Also, make use proper exception handling by using inbuilt exception-classes and also make your own custom exception class to deal with the exceptions such as user entering a negative no.**

6) Write a program to implement following scenario:
   Create an interface **Color** having:

| | | |
|---|---|---|
| **default** method: | void showColor() | → displays the color of object |
| **abstract** methods: | double area() | → will be used to return area of object |
| | double perimeter() | → will be used to return perimeter of object |

   Now, create a class **Rectangle** which inherits **Color** interface and has:
   **private** attributes: length, width
   Make setter and getter methods for these attributes.

   Now, create a class **Test** having main() method in which you have to make “n” objects of Rectangle class (where “n” is specified by user at run-time). Also, the required details about the rectangle objects have to be taken as input from the user during run-time.
   Make a method named: saveRecords() in the same class which will perform the task of Serialization of only those rectangle-objects whose area exceeds value of 100.
   Make another method named: showRecords() in the same class which will perform the task of de-serialization and display complete information of all rectangle objects.

   **Note: Don’t make any separate instance variable other than the ones specified in above scenario. Also, make use proper exception handling by using inbuilt exception-classes and**

**also make your own custom exception class to deal with the exceptions such as user entering a negative no.**