**OOAD using Java : Week – 3 Hands on session**

**Problem to be solved – 1:**

Write a Java program to implement a pile of cards as a **stack data structure**.

A **pile** contains cards where you can place a card on the top of the pile (push) and draw a card from the top of the pile (pop).

A **card** should be an Object with properties card suit (Club, Diamond, Spade or Heart) and a card value (A, 1, 2, 3…10, J, Q, K).

You can also peek in to the pile by checking the card at the top without drawing it from the pile (display). The pile can contain a maximum of 10 cards only. The pile should be an object instantiated in the main function. After that the user should be given an option to perform any of the three functions (place, draw or peek).

The client code looks as below. Write the implementations accordingly.

class StackDemo {

public static void main(String[] args) {

Pile stack = new Pile();

int ch;

System.out.println("enter your choice 1. place 2.draw 3.peek 4. exit");

Scanner input = new Scanner(System.in);

ch = input.nextInt();

do

{ switch(ch)

{

case 1: System.out.println("enter the card suit and card value");

String suit = input.next();

String value = input.next();

Card c = new Card(suit, value);

stack.place(c);

break;

case 2: Card d = stack.draw();

if(d.value != null)

System.out.println("Card Drawn:" + d.display());break;

case 3: stack.peek();break;

default: System.exit(0);

}

System.out.println("enter your choice 1. place 2.draw 3.peek 4. exit");

ch = input.nextInt();

}while(ch<4);

}

}

**Problem to be solved – 2:**

Write a Java program to implement the following.

1. Define an abstract class TestQuestion that has a String data variable called question and a readQuestion abstract method.

2. Define three subclasses ShortAnswer, LongAnswer and MCQ. The subclasses should have the following data variables in addition to the question, ShortAnswer (numLines – by default set to 1), LongAnswer (numLines) and MCQ(numChoices, array of String for the choices) 3. The three subclasses define the readQuestion method as follows:

a. ShortAnswer would read the question from standard input (keyboard) and also sets the numLines to 1 by default.

b. LongAnswer would read the question and numLines from standard input (keyboard)

c. MCQ would read the question, numChoices and choices from standard input (keyboard)

4. Write the toString method for each of the subclasses to display the details

5. The main method in TQManager class should contain an array of TestQuestions that references any type of subclasses. In the main function, the user chooses to create a question of a specific type and accordingly an instance is created and a reference is assigned in the array and the readQuestion method is invoked. Thereafter, display all the questions by implicitly invoking the toString method.

**Additional questions:**

1. Implement a car parking waiting line as a queue data structure. You can add a car (Reg. No., Make) to the end of the waiting line. Once a car parking spot is available the car in the front of the line is removed from the waiting line. The waiting line can hold a maximum of 10 cars. Implement the queue using arrays.

2. Write an integer list class. It should create a list of a fixed size and have a method add to add an element to the end of a list. If the list is already full, a message will be printed. Now write a class SortedIntList that extends IntList. SortedIntList should be just like IntList except that its elements should always be in sorted order from smallest to largest. This means that when an element is inserted into a SortedIntList it should be put into its sorted place, not just at the end of the array. Hence, you need to override the add method. (Optionally, you can use recursion to locate the position of the element to be inserted.