Université de BATNA
Faculté MI
Département d'Informatique

Filière : Master 1 IRC
Module : Technologie Objet

Travaux Dirigés N°1 (Rappel)

Exercice 1 :

Cette étude de cas concerne un système simplifié de réservation de vols dans une agence de voyage. Les interviews ont permis de faire ressortir les règles de gestion décrivant la réalité :

1- Des compagnies aériennes proposent différents vols.
2- Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.
3- Un client peut réserver un ou plusieurs vols, pour des passagers différents.
4- Une réservation concerne un seul vol, et un seul passager.
5- Une réservation peut être annulée ou confirmée.
6- Un vol a un aéroport de départ et un aéroport d'arrivée.
7- Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.
8- Un vol peut comporter des escales dans des aéroports
9- Une escale a une heure d'arrivée et une heure de départ.
10- Chaque aéroport dessert une ou plusieurs villes

I – Après analyse, concevoir le diagramme de classe correspondant à la description.

II- Elaborer le diagramme objet relatif à l'itinéraire d'un vol qui fait Batna-Paris avec une escale à Alger.

Exercice 2 :

La pile est un TAD (Type Abstrait de Donnée) qui peut être implémentée sous formes diverses : Tableau, Liste chainée,… Les éléments de la pile peuvent être de différentes nature et type : Générique.

1- Donner la spécification du TAD Pile.
2- Conformément à la spécification, donner la classe Pile adéquate.
3- On suppose que la représentation concrète de la pile est un tableau d'entiers. Donner le code en Java et en C++ de la classe Pile.
4- L'un des éléments paradigmatique de l'approche orientée objet est la Généricité. En adoptant ce paradigme, proposer un code java d'une pile générique.
5- Reprendre les mêmes étapes avec le TAD File.

Exercice 2 :

1- La spécification du TAD Pile

Types

Stack(Element)

Functions

New : ( ) → Stack

Push : Stack X Element → Stack

Pop : Stack → Stack

Empty : Stack → Boolean

Peek : Stack → Element

Preconditions

Pop(s) : ¬Empty(s)

Peek(s) : ¬Empty(s)

Axiomes

Empty(New()) = True

Empty(Push(s,e)) = False

Pop(Push(s,e)) = s

Peek(Push(s,e)) = e

2- Classe Pile d'entier en Java

```
public class Stack {
  private int MaxSize=20;
  private int top;
  private int[] elements;

  public Stack(){
    top=-1;
    elements = new int[MaxSize];
  }

  public Stack(int max){
    top=-1;
    MaxSize=max;
    elements = new int[MaxSize];
  }

  public boolean Empty() {
    return (top==-1);
  }

  public void Push(int value) {
    if (top<MaxSize-1) {
      top++;
      elements[top]=value;
    } else {
      System.out.println("Stack
Overflow !");
    }
  }
```

```
  public int Pop() {
    int v=-1  ;

    if (!Empty()) {
      v=elements[top];
      top--;
    } else {
      System.out.println("Stack
Empty !");
    }
    return v;
  }

  public int Peek() {
    if (!Empty()) {
      return elements[top];
    } else {
      return -1;
    }
  }
}
```

```
public class StackTest {

  public static void main(String args []) {
    Stack myStack = new Stack(3);
    myStack.Push(8);
    System.out.println(myStack.Peek());
    myStack.Push(10);
    System.out.println(myStack.Peek());
    myStack.Push(12);
    System.out.println(myStack.Peek());
    myStack.Push(15);
    System.out.println(myStack.Peek());
    myStack.Pop();
    myStack.Pop();
    System.out.println(myStack.Peek());
    myStack.Pop();
    System.out.println(myStack.Peek());
    myStack.Pop();
    System.out.println(myStack.Peek());
  }
}
```