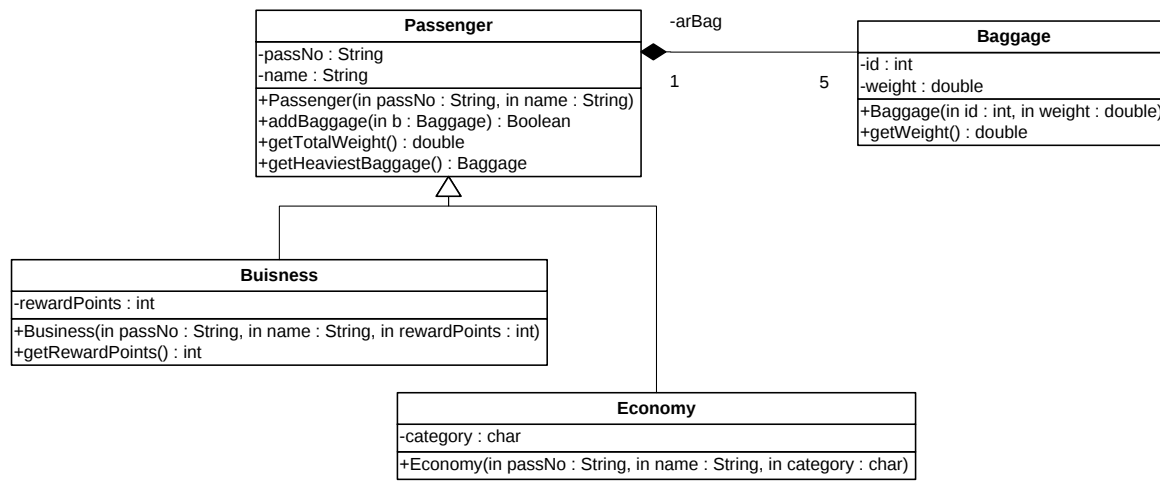


King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Final Exam – Spring 2016

Exercise1:



Baggage class:

- Attributes:
 - ***id***: the id of the baggage item.
 - ***weight***: the weight of the baggage item.
- Methods:
 - ***Baggage(id: int, weight: double)***: constructor.
 - ***getWeight()***: this method returns the weight of the baggage item.

Passenger class:

- Attributes:
 - ***passNo***: the passport number of the passenger.
 - ***name***: the name of the passenger.
- Methods:
 - ***Passenger(passNo: String, name: String)***: constructor.
 - ***addBaggage(b: Baggage)***: this method adds the Baggage ***b*** to the passenger. It returns true if ***b*** is added successfully, and false otherwise.
 - ***getTotalWeight()***: this method calculates and returns the total weight of all baggage of the passenger.

- ***getHeaviestBaggage()***: this method returns the baggage object that has the maximum weight among all baggage of the passenger.

Business class:

- Attributes:
 - ***rewardPoints***: the number of reward points of the business passenger.
- Methods:
 - ***Business (passNo: String, name: String, rewardPoints: int)***: constructor
 - ***getRewardpoints()***: this method returns the reward points of the business passenger.

Economy class

- Attributes:
 - ***category***: the category of the economy passenger.
- Methods:
 - ***Economy (passNo: String, name: String, category: char)***: constructor

QUESTION: Translate into Java code:

- the class ***Baggage***
- and the class ***Passenger***.

Answer:

```
public class Baggage { ..... /3
    private int id;
    private double weight;

    public Baggage(int id, double weight){
        this.id = id;
        this.weight = weight;
    }

    public Baggage(Baggage b){ ..... 2
        id = b.id;
        weight = b.weight;
    }

    public double getWeight(){
        return weight; ..... 1
    }
}
```

```

public class Passenger { ..... / 18
    private String passNo;
    private String name;

    private Baggage[] arBag; ..... 1
    private int nbBag; ..... 1

    public Passenger(String passNo, String name){
        this.passNo = passNo;
        this.name = name;

        arBag = new Baggage[5]; ..... 1
        nbBag = 0; ..... 1
    }

    public boolean addBaggage(Baggage b){
        if (nbBag < arBag.length) { ..... 1
            arBag[nbBag] = new Baggage(b); ..... 1
            nbBag++; ..... 1
            return true; ..... 0.5
        }
        else
            return false; ..... 0.5
    }

    public double getTotalWeight(){
        double total = 0; ..... 1

        for (int i=0; i < nbBag; i++) ..... 1
            total += arBag[i].getWeight(); ..... 1

        return total; ..... 1
    }

    public Baggage getHeaviestBaggage(){
        if (nbBag == 0) return null; ..... 0.

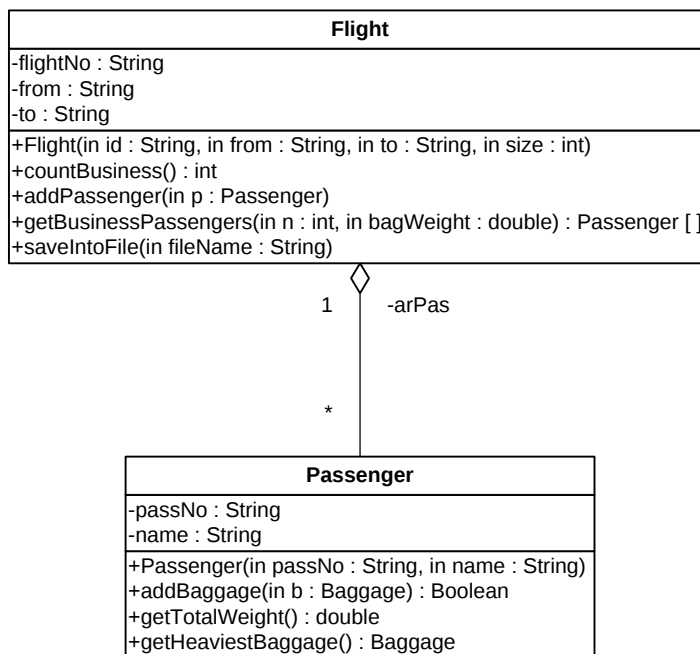
        Baggage result = arBag[0]; ..... 1
        for (int i=1; i < nbBag; i++) ..... 1
            if (arBag[i].getWeight() > result.getWeight()) ..... 1
                result = arBag[i]; ..... 1

        return result; ..... 1
    }
}

```

Exercise 2:

Let's consider the same class *Passenger* described in exercise 1.



Flight class:

- Attributes:
 - **flightNo**: the flight number.
 - **from**: the name of the departure airport.
 - **to**: the name of the arrival airport.
- Methods:
 - **Flight (id: String, from: String, to: String, size: int)**: constructor. The parameter **size** defines the maximum number of passengers in the flight.
 - **countBusiness ()**: this method returns the number of business passengers in the flight.
 - **addPassenger (p: Passenger)**: this method adds the passenger **p** to the flight *if possible*. There are exactly 10 seats for business passengers on each flight. If adding a passenger is not possible, this method raises an exception with the following message “**No available seats**”.
 - **getBusinessPassengers(n: int, bagWeight: double)**: this method returns an array containing all Business passengers having reward points less than **n**, and total baggage weight exceeding **bagWeight**.

- **saveIntoFile(filename: String):** this method stores all passenger objects of the flight in a file named *filename*.

QUESTION: Translate into Java code the class *Flight*.

Answer:

```
public class Flight { ..... / 33
    private String flightNo, from, to;

    private Passenger[] arPas; ..... 1
    private int nbSeats; ..... 1

    public Flight(String id, String from, String to, int size){
        flightNo = id;    this.from = from;    this.to = to;
        arPas = new Passenger[size]; ..... 1
        nbSeats = 0; ..... 1
    }

    public int countBusiness(){
        int n = 0; ..... 1

        for (int i=0; i < nbSeats; i++) ..... 1
            if (arPas[i] instanceof Business) ..... 1
                n++; ..... 1

        return n; ..... 1
    }

    public void addPassenger(Passenger p) throws Exception{ ..... 1
        if (nbSeats == arPas.length)
            throw new Exception("No available seats");

        if (p instanceof Business && countBusiness() == 10) ..... 1+1
            throw new Exception("No available seats"); ..... 0.5

        if (p instanceof Economy && ..... 1
            ((nbSeats - countBusiness()) == (arPas.length - 10))) ..... 1
            throw new Exception("No available seats"); ..... 0.5

        arPas[nbSeats++] = p; ..... 1+1
    }

    public Passenger[] getBusinessPassengers(int n, double bagWeight){
        int nbB = countBusiness();
        Passenger[] result = new Passenger[nbB]; ..... 1+1
        int j = 0; ..... 1
```

```

    for (int i=0; i < nbSeats; i++) ..... 1
        if (arPas[i] instanceof Business && ..... 1
            arPas[i].getTotalWeight() >= bagWeight && ..... 1
            ((Business)arPas[i]).getRewardPoints() < n) ..... 1+ 1
                result[j++] = arPas[i]; ..... 1+ 1
    return result; ..... 1
}

public void saveIntoFile(String fileName) throws Exception{ ..... 0.5
    File f = new File(fileName); ..... 0.5
    FileOutputStream fos = new FileOutputStream(f); ..... 0.5
    ObjectOutputStream oos = new ObjectOutputStream(fos); ..... 0.5

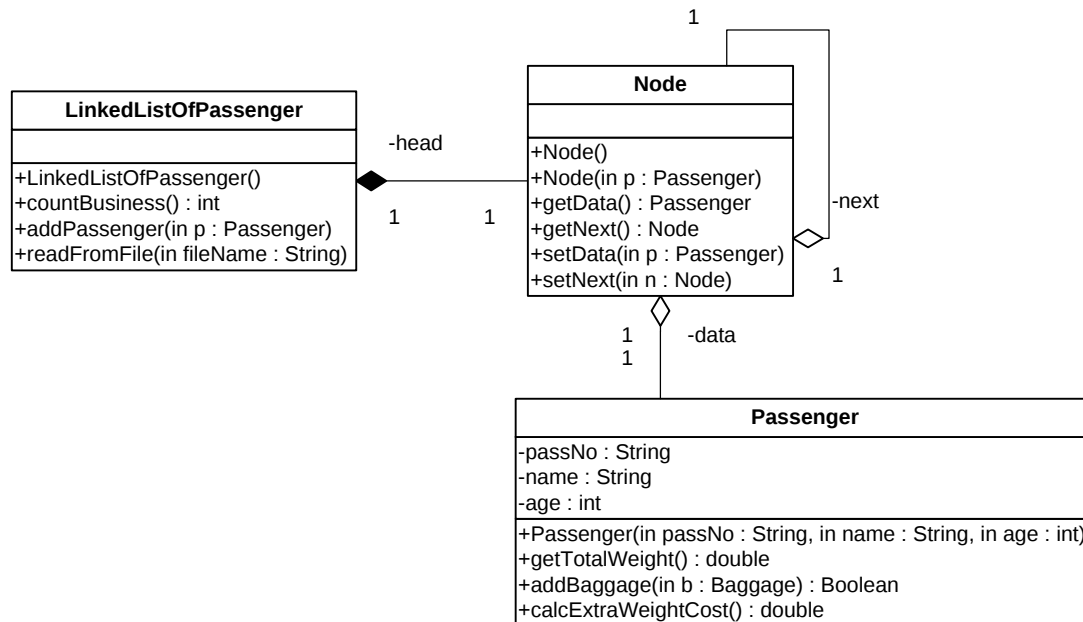
    for (int i=0; i < nbSeats; i++) ..... 1
        oos.writeObject(arPas[i]); ..... 1

    oos.close(); ..... 1
}
}

```

Exercise 3:

Let's consider the same class *Passenger* described in exercise 1.



LinkedListOfPassenger class:

○ Methods:

- **LinkedListOfPassenger()**: constructor.
- **countBusiness()**: this method returns the number of business passengers in the list.
- **addpassenger(p: Passenger)**: this method inserts the passenger *p* at the back of the list.
- **readFromFile(filename: String)**: this method reads all passenger objects stored in the file named *filename* and adds them to the list.

QUESTION: Translate into Java code the class *LinkedListOfPassenger*.

Answer:

```
public class LinkedListOfPassenger { ..... /26
    private Node head; ..... 1

    public LinkedListOfPassenger() {
        head = null; ..... 1
    }

    public int countBusiness() {
        int n = 0; ..... 1
        Node p = head; ..... 1

        while(p != null) { ..... 1
            if (p.getData() instanceof Business) ..... 1
                n++; ..... 1

            p = p.getNext(); ..... 1
        }
        return n; ..... 1
    }

    public void addPassenger(Passenger p) {
        Node nn = new Node(p); ..... 1
        //Node nn = new Node(); nn.setData(p);

        if (head == null) ..... 1
            head = nn; ..... 1
        else {
            Node tail = head; ..... 1
            while (tail.getNext() != null) ..... 1
                tail = tail.getNext(); ..... 1

            tail.setNext(nn); ..... 1
        }
    }

    public void readFromFile(String fileName) throws Exception{ ..... 0.5
        File f = new File(fileName); ..... 0.5
        FileInputStream fis = new FileInputStream(f); ..... 1
        ObjectInputStream ois = new ObjectInputStream(fis); ..... 1

        try { ..... 1
            while (true) { ..... 1
                Passenger p = (Passenger)ois.readObject(); ..... 1 + 1
                addPassenger(p); ..... 1
            }
        } catch (EOFException e) {} ..... 1
        ois.close(); ..... 1
    }
}
```

}