# Objective

Practice polymorphism, array of objects, Object class and interfaces

# Creativity

Feel free to be creative by adding more functionalities. Make sure that the required functionalities are not changed.

# Problem

Southwest has asked us to create an application to help them keep track of the list of their **passengers** in each **plane**. They also want to be able to do the following to list of the passengers

* Display the information of all the passengers in the plane
* Delete a passenger from the list in a plane
* Add a passenger to the list of the plane if the plane hasn’t taken off
* Count the number of the passengers in the plane
* Search the list of the passengers to find a particular person in the plane
* display the last name of all the passengers

A **passenger** has the following attributes

* first name
* Last name
* Ticket ID
* Seat number
* class (first class, business class, premium economy, economy)

**Plane**

A plane has a list of the passengers. **Therefore, plane class does not extend the passenger class**

# Design

In order to apply the OOP, we will need to create the following user defined data types

* Person class
* Passenger class (A passenger is a person)
* Plane class (a plane has a list of the passengers)
* List interface

# Classes descriptions

1. **Person class** with the attributes, name, last, and ticketID. This class will have the constructor, getter methods, setter methods, equals method and toString method. The method header for the equals method must be: **public boolean equals (Object o).** In this method you must use **instanceof** keyword to make sure that the parameter **o** passed to this method is of type Person, then type cast it to the Person class. Then compare the last names of the two objects. If the last names are the same then you must compare the two passengers based on their first name.

|  |
| --- |
| **Person class** |
| -name: String  -last name: String  -ticketID: String |
| +Person (String name, String last, String ticketID)  +setFirst(String name): void  +setLast(String las): void  +setTicketID (String TicketID): void  +getFirst(): String  +getLast(): String  +getTicketID(): String  +toString(): String  +equals(Object o): boolean// compare the first and the last names |

1. **Passenger class** extends the **person** class since a passenger is a person. A Passenger has the following attributes: seat number, class (first class, business class, premium economy, economy). This class will have the constructor, getter methods, setter methods, toString method, remember to call the toString methods from the parent class in the toString method in this class. There is no need to override the equals method from the Person class.

|  |
| --- |
| Passenger class //extends the person class |
| -seatNumber: int  -classType :String |
| +Passenger(String first, String last, String ticketID, int seatNumber, String classType)  +setSeatNumber(int num): void  +setClass(String classType): void  +getClassType(): String  +getSeatNumber() : int  +toString(): String // must call the tostring method from the person class |

1. List interface

copy and paste the following interface into your program

interface list

{

public boolean add(Object o)

public Object search(Object o);  
public boolean delete(Object o);  
public void printLast();

public void takeOff();

}

1. **Airplane class must implement the List interface.** Airplane classhas a list of the passengers (has -a relationship between the Airplane class and the passenger class).
   1. Therefore, the attribute for this class is an array of passengers (Passenger [] plane).
   2. This class should have a class variable to count the number of the passengers in the plane. Every time a passenger is added to this plane, the count must be incremented. Every time a passenger is deleted from the list, the count must be decremented by 1.
   3. This class must have a constructor to instantiate the array of passengers. You should initiate the array to a size of 10, set takenOff to false and initiate the variable planeNum in the constructor
   4. Implement the toString method by using a for loop traversing the list(array) of the passengers, call the tostring method on each element of the array plane[i].toString() in the for loop.
   5. getCount method is a static method. This method returns the number of the passengers in the plane. to call this method from the driver class: plane.getCount()
   6. This class implements the List interface providing the body for the methods add, search, delete, and printLast.

|  |
| --- |
| Airplane |
| -public static int count = 0;  -plane: Passenger[];  -takenOff: boolean  -planeNum: int |
| +Airplane(int planeNum)  +getPlaneNumber(): int  +setPlaneNumber(int num): void  +toString(): String  +public static int getCount(): int  //methods form the interface |

# Description of the methods listed in the List interface that must be implemented in the Trian class

**public boolean add(Object o):** This method gets the Object o as its parameter. This method must check the class type of the Object o by using the keyword instanceof. if the type is of type Passenger, type cast it to the Passenger class and check if the plane has taken off then add it to the array. Every time a Passenger is added to the list, the class variable count must be incremented. don’t forget to increment the count by 1.

**public Object search (Object o**: This method searches the list of the passenger to find the passenger with the particular last name. this method gets the Object o as its parameter . However, the Object o represents the passenger’s last name therefore using the keyword instanceof would check the Class type of the Object o to see if it is of string type, if it is then type cast o to String class. Then search the array using the last name. if the last name is found return the object at the given index. Here is the code for this method;

public Object search (Object o)

{

boolean b = o instanceof String;

if(!b)

return null;  
 String name = (String)o;

for(int i = 0; i < plane.length; i++)  
 {  
 if(plane[i]!= null && plane[i].getLast().equalsIgnoreCase(name))  
 { return plane[i]; //returning the found object }

}  
return null;//returning null if the object is not found

}  
public boolean delete(Object o): this method is very similar to the search method. Once you find the passenger in the list, delete it from the list by setting the element at the given index to null and return true. If the passenger is not found return false. don’t forget to decrement the count by 1 once you delete a passenger. .

public void printLast(): this method uses a for loop to display the last name of the passengers in the plane. Method getLast from the person class must be called. this line of code should be used in the for loop.: plane[i].getLast() . The rest of the code is similar to the search method

public void takeOff(): this method sets the variable takenOff to true. This should disable the ability to add Passengers to an Airplane that has taken off.

## Driver class

1. Some code has been provided for this class in the given shell. Once you finish implementing all the classes uncomment the given driver class. If you have implemented everything correctly you should see the given sample output.
2. **Add your own code to the driver class and run it again.**

* Instantiate an object of the Airplane class: Airplane Plane2 = new Airplane((plane number))
  + Create 5 objects of type passengers of your own choice, name the objects mom, dad, sister, brother and uncle: Here is a sample of a passenger. Passenger dad = new Passenger ("Nick", "Rami","123456789”, 12, "First class");
  + Add the passengers mom, dad, brother, sister and uncle to Plane2 by calling the add method: Plane2.add(dad)
  + Display the last name of all the passengers (Plane2.printLast())
  + Prompt the user for a last name to search the list of the passengers. If the passenger found in the list, display their info. You need to call the search method
  + Prompt the user for a last name to delete it from the list of the passengers. Call the delete method to remove the given person from the sierraTrain
  + Call the toString method to display the updated list (System.out.println(sierraTrain))
  + Have the plane take off using the takeOff() method.
  + Create a new Passenger and add it to Plane2.

# sample output for the given code in the driver:



