**Row Number**

The flight has landed at the Karachi International Airport. Most of the passengers from Karachi were travelling by air for the first time. Neerja, the purser wanted to help the passengers locate their seats.



The Pan Am aircraft had R rows with C seats numbered from 1. Given a seat number can you determine the row number in which the seat would fall in.  
  
**Note** : Rows and columns are numbered starting from 1.  
  
**Input Format :**  
The first line of input is an integer R corresponding to the number of rows.  
The second line of input is an integer C corresponding to the number of seats in a row.  
The third line of input is an integer S corresponding to the seat number.  
  
  
**Output Format :**  
The output consists of one line.  
Print an integer corresponding to the row number in which the seat falls if the seat number exists.  
Print “Invalid Input” if the seat number does not exist.  
Refer sample input(s) and output(s) for formatting specifications.  
  
**Sample Input 1:**  
10  
20  
11  
**Sample Output 1:**  
1  
  
**Sample Input 2:**  
-9  
20  
13  
**Sample Output 2:**  
Invalid Input

Top of Form

Bottom of Form

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int R=sc.nextInt();

int C=sc.nextInt();

int S=sc.nextInt();

if(R>0&&C>0&&S>0)

{

System.out.println(S%R);

}

else

{

System.out.println("Invalid Input");

}

}

}

**Inheritance - Aircraft Seat- Business Class Seat**

A passenger can travel in flight in the normal aircraft seat or the Business class seat. Business class of service offers significantly more comfort and amenities than standard Economy. While in-flight amenities, service and entertainment available in Business Class are often comparable across airline carriers, the type of seat offered can vary significantly. Business Class seats can be categorized by one of the following types:

1.Recliner Seats

2.Angle Lie-Flat Seats

3.Flat Bed Seats

4.Suites

They are often provided with a Larger personal TV screens and more viewing options that could provide news, sports and entertainment channels.

Create a class **AircraftSeat** with the following private variables:

|  |  |
| --- | --- |
| **Variable Name** | **Data Type** |
| String | aircraftName |
| Int | seatId |
| String | travelClassid |

Include getters and setters method in the AircraftSeat class.

Create a class **BusinessClassSeat**(which should inherit AircraftSeat)  with the following private variables:

|  |  |
| --- | --- |
| **Variable Name** | **Data Type** |
| int | businessClassSeatid |
| String | seatType |
| String | tvChannelpreferences |

Include getters and setters method in the BusinessClassSeat class.

Create a method  void displayDetails() in the BusinessClassSeat class. This method should display the client details along with the agent details.

**Sample input and output:**

Enter the aircraft name

**lufthansa**

Enter the seat id

**24**

Enter the travelClassid

**B**

Enter the seat Type

**Recliner seat**

Enter the businessClassSeatid

**12**

Enter the TV Channel preferences

**Entertainment**

Aircraft details

Aircraft name : lufthansa

Seat Id : 24

Travelclass id : B

Seat type : Recliner seat

Business Class seat id : B12

TV Channel preferences : Entertainment

Top of Form

Bottom of Form

public class AircraftSeat {

private String aircraftName;

private int seatId;

private String travelClassid;

public String getAircraftName() {

return aircraftName;

}

public void setAircraftName(String aircraftName) {

this.aircraftName = aircraftName;

}

public int getSeatId() {

return seatId;

}

public void setSeatId(int seatId) {

this.seatId = seatId;

}

public String getTravelClassid() {

return travelClassid;

}

public void setTravelClassid(String travelClassid) {

this.travelClassid = travelClassid;

}

}

public class BusinessClassSeat extends AircraftSeat {

private int businessClassSeatid;

private String seatType;

private String tvChannelpreferences;

public int getBusinessClassSeatid() {

return businessClassSeatid;

}

public void setBusinessClassSeatid(int businessClassSeatid) {

this.businessClassSeatid = businessClassSeatid;

}

public String getSeatType() {

return seatType;

}

public void setSeatType(String seatType) {

this.seatType = seatType;

}

public String getTvChannelpreferences() {

return tvChannelpreferences;

}

public void setTvChannelpreferences(String tvChannelpreferences) {

this.tvChannelpreferences = tvChannelpreferences;

}

public void displayDetails()

{ System.out.println("Aircraft details ");

System.out.println("Aircraft name : "+getAircraftName());

System.out.println("Seat Id : "+getSeatId());

System.out.println("Travelclass id : "+getTravelClassid());

System.out.println("Seat type : "+getSeatType());

System.out.println("Business Class seat id : "+getTravelClassid()+getBusinessClassSeatid());

System.out.println("TV Channel preferences : "+getTvChannelpreferences());

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static void main(String args[]) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

BusinessClassSeat c = new BusinessClassSeat();

System.out.println("Enter the aircraft name");

c.setAircraftName(br.readLine());

System.out.println("Enter the seat id");

int seatId=Integer.parseInt(br.readLine());

c.setSeatId(seatId);

System.out.println("Enter the travelClassid");

c.setTravelClassid(br.readLine());

System.out.println("Enter the seat Type");

c.setSeatType(br.readLine());

System.out.println("Enter the businessClassSeatid");

int businessClassSeatid=Integer.parseInt(br.readLine());

c.setBusinessClassSeatid(businessClassSeatid);

System.out.println("Enter the TV Channel preferences");

c.setTvChannelpreferences(br.readLine());

c.displayDetails();

}

}

**Abstract Classes - Booking Tickets**

[Adhere to the OOPs specifications specified here. Follow the naming conventions for getters and setters.]

Create an abstract class named Aircraft with the following private attributes / member variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | aircraftName |
| String | Source |
| String | Destination |

 Include appropriate getters and setters.

Include a 3-argument constructor, the order of the arguments is aircraftName, source,destination.

Include an abstract method named displayDetails(), The return type of this method is void.

Create a class named **PublicAircraft** . The class PublicAircraft  is a derived class of Booking. Include the following private attributes / member variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Boolean | checkinbeforetwohours |
| int | noOfKgsallowed |
| float | additionalFeeperkg |

Here checkinbeforetwohours is a Boolean value that says whether the passenger should check in before two hours or not.

 This flag is false in case of PrivateAircraft.

 Include appropriate getters and setters.

Include a 6-argument constructor, the order of the arguments is aircraftName, source,destination, checkinbeforetwohours, pilotPreference, purpose

Override the abstract method displayDetails() defined in the Aircraft class, this method prints the booking details as entered by the passenger (refer sample input and output).

Create a class named **PrivateAircraft** . The class PrivateAircraft  is a derived class of Booking. Include the following private attributes / member variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Boolean | checkinbeforetwohours |
| String | pilotPreference |
| String | purpose |

 Include appropriate getters and setters.

Include a 6-argument constructor, the order of the arguments is aircraftName, source,destination, checkinbeforetwohours, pilotPreference., purpose

Override the abstract method displayDetails() defined in the Aircraft class, this method prints the booking details as entered by the passenger (refer sample input and output).

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input Output 1:**

Enter the name of the Aircraft

**Jet Airways**

Enter the source

**Bangalore**

Enter the destination

**Chennai**

Enter the type of Flight

1.Public Aircraft

2.Private Aircraft

**1**

Is the flight check in before two hours

**Yes**

Enter the number of kgs allowed per person

**15**

Enter the additional fee charged for extra baggage per Kg

**750.00**

Flight Details :

Private Aircraft

Aircraft Name : Jet Airways

Source : Bangalore

Destination : Chennai  
Flight check in before two hours : Yes

Number of kgs allowed per person : 15

Additional fee charged for extra baggage per Kg : 750.00  
  
**Sample Input and Output 2 :**

Enter the name of the Aircraft

**Jet Airways**

Enter the source

**Chennai**

Enter the destination

**Bangalore**

Enter the type of Flight

1.Public Aircraft

2.Private Aircraft

**2**

Is the flight check in before two hours

**No**

Enter the name of the pilot choosed

**Akilan**

Enter the Purpose of your flight

**Medical**

Flight Details :

Private Aircraft:

Aircraft Name : Jet Airways

Source : Chennai

Destination : Bangalore

Flight check in before two hours : No

Pilot choosed : Akilan

Purpose of the flight :  Medical

Top of Form

Bottom of Form

public abstract class Aircraft {

private String aircraftName;

private String source;

private String destination;

public Aircraft(String aircraftName, String source, String destination) {

super();

this.aircraftName = aircraftName;

this.source = source;

this.destination = destination;

}

public String getAircraftName() {

return aircraftName;

}

public void setAircraftName(String aircraftName) {

this.aircraftName = aircraftName;

}

public String getSource() {

return source;

}

public void setSource(String source) {

this.source = source;

}

public String getDestination() {

return destination;

}

public void setDestination(String destination) {

this.destination = destination;

}

@Override

public String toString() {

return "Aircraft [aircraftName=" + aircraftName + ", source=" + source + ", destination=" + destination

+ ", getAircraftName()=" + getAircraftName() + ", getSource()=" + getSource() + ", getDestination()="

+ getDestination() + ", getClass()=" + getClass() + ", hashCode()=" + hashCode() + ", toString()="

+ super.toString() + "]";

}

public abstract void displayDetails();

}

public class PrivateAircraft extends Aircraft {

private Boolean checkinbeforetwohours;

private String pilotPreference;

private String purpose;

public PrivateAircraft(String aircraftName, String source, String destination, Boolean checkinbeforetwohours,

String pilotPreference, String purpose) {

super(aircraftName, source, destination);

this.checkinbeforetwohours = checkinbeforetwohours;

this.pilotPreference = pilotPreference;

this.purpose = purpose;

}

public Boolean getCheckinbeforetwohours() {

return checkinbeforetwohours;

}

public void setCheckinbeforetwohours(Boolean checkinbeforetwohours) {

this.checkinbeforetwohours = checkinbeforetwohours;

}

public String getPilotPreference() {

return pilotPreference;

}

public void setPilotPreference(String pilotPreference) {

this.pilotPreference = pilotPreference;

}

public String getPurpose() {

return purpose;

}

public void setPurpose(String purpose) {

this.purpose = purpose;

}

@Override

public void displayDetails() {

System.out.println("Aircraft Name : "+getAircraftName());

System.out.println("Source : "+getSource());

System.out.println("Destination : "+getDestination());

System.out.println("Flight check in before two hours :"+getCheckinbeforetwohours());

System.out.println("Pilot choosed :"+getPilotPreference() );

System.out.println("Purpose of the flight : "+getPurpose());

}

}

public class PublicAircraft extends Aircraft {

private Boolean checkinbeforetwohours;

private int noOfKgsallowed;

private float additionalFeeperkg;

public PublicAircraft(String aircraftName, String source, String destination,

int noOfKgsallowed, float additionalFeeperkg,Boolean b1) {

super(aircraftName, source, destination);

this.checkinbeforetwohours = checkinbeforetwohours;

this.noOfKgsallowed = noOfKgsallowed;

this.additionalFeeperkg = additionalFeeperkg;

}

public Boolean getCheckinbeforetwohours() {

return checkinbeforetwohours;

}

public void setCheckinbeforetwohours(Boolean checkinbeforetwohours) {

this.checkinbeforetwohours = checkinbeforetwohours;

}

public int getNoOfKgsallowed() {

return noOfKgsallowed;

}

public void setNoOfKgsallowed(int noOfKgsallowed) {

this.noOfKgsallowed = noOfKgsallowed;

}

public float getAdditionalFeeperkg() {

return additionalFeeperkg;

}

public void setAdditionalFeeperkg(float additionalFeeperkg) {

this.additionalFeeperkg = additionalFeeperkg;

}

@Override

public void displayDetails() {

System.out.println("Aircraft Name : "+getAircraftName());

System.out.println("Source : "+getSource());

System.out.println("Destination : "+getDestination());

System.out.println("Flight check in before two hours :"+checkinbeforetwohours);

System.out.println("Number of kgs allowed per person : "+getNoOfKgsallowed() );

System.out.println("Additional fee charged for extra baggage per Kg : "+getAdditionalFeeperkg());

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static void main(String args[]) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the name of the Aircraft");

String aircraftName=br.readLine();

System.out.println("Enter the source");

String source=br.readLine();

System.out.println("Enter the destination");

String destination = br.readLine();

System.out.println("Enter the type of Flight\n1.Public Aircraft\n2.Private Aircraft");

int choice=Integer.parseInt(br.readLine());

if(choice==1)

{

System.out.println("Is the flight check in before two hours");

String ans = br.readLine();

Boolean b1;

if(ans.equals("yes"))

{

b1=true;

}

else

{

b1=false;

}

System.out.println("Enter the number of kgs allowed per person");

int noOfKgsallowed=Integer.parseInt(br.readLine());

System.out.println("Enter the additional fee charged for extra baggage per Kg");

float additionalFeeperkg=Float.parseFloat(br.readLine());

Aircraft b = new PublicAircraft(aircraftName,source,destination,noOfKgsallowed,additionalFeeperkg,b1);

System.out.println("Flight Details :");

System.out.println("Public Aircraft:");

b.displayDetails();

}

if(choice==2)

{

System.out.println("Is the flight check in before two hours");

Boolean ans = Boolean.parseBoolean(br.readLine());

System.out.println("Enter the name of the pilot choosed");

String pilotPreference=br.readLine();

System.out.println("Enter the Purpose of your flight");

String purpose=br.readLine();

Aircraft b = new PrivateAircraft(aircraftName,source,destination,ans,pilotPreference,purpose);

System.out.println("Flight Details :");

System.out.println("Private Aircraft:");

b.displayDetails();

}

}

}

**Extra Passengers list**

Air India allows the passengers to book the tickets for festive time with special offers. Due to the Increased booking to travel from Bangalore to Chennai, the Air craft that could accomodate larger number of passengers  which was alloted for the passengers travelling from Chennai to Coimbatore has been rescheduled to Bangalore to Chennai location. So the passengers travelling from Chennai to Coimbatore were alloted with a smaller size Air craft. Write a program to find out the passengers who has been left out without seats in Chennai to Coimbatore travel.   
Create a list that holds the booked passenger names. Display the passengers who has been left out without seats.

**Input and Output Format:**  
[ Refer sample input and output for formatting specifications.  
All text in bold corresponds to input and the rest corresponds to output. ]  
  
  
**Sample Input and Output :**  
  
Enter the number of passengers Booked  
**8**  
Enter the passenger's name who Booked  
**Thara  
Shanmathi  
Raghul  
Haritha  
madhu  
Praveen  
Jimesh  
Krishna**  
Enter the number of seats available  
**5**  
Extra Passengers list  
[Praveen, Jimesh, Krishna]

Top of Form

Bottom of Form

**InvalidBookException**

**[Note : Strictly adhere to the object-oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]**   
  
Create class **Book**with following variables/ attributes.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | Code |
| String | Title |
| String | Author |
| double | Price |

Include appropriate **getters**and **setters**.  
Include **default**and **parameterized constructor**with parameters in the following order  
**public Book(String code, String title, String author, String price)**  
Override **toString()** method to display the book details.

Create a class named as **BookBO**, which contains following methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| public Book createBook(String details) | In this method, create a book object. Validate the book code with the following conditions. 1. The book code should always start with the starting character of the book name (case insensitive). 2. Next two character should be equal to the first two characters of author name(case insensitive). 3. Next set of characters should be integers.  If the book code fails any condition throw **InvalidBookException**, else create and return the book object. |

Create a custom exception class **InvalidBookException** which extends Exception class.  
  
Create a driver class **Main**with the **main** method to test the above classes. Get the book details from user in comma separated format. Create book object using createBook() method and display the book details using toString() method in Book class. Handle exception using try catch and display the exception message.

**Input and Output Format**  
Refer sample input and output for formatting specifications.  
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1:**

Enter the book details(code,title,author,price)  
**CMA001,Clean Code,Martin,2000**  
Book details:   
Code: CMA001  
Title: Clean Code  
Author: Martin  
Price: 2000.00

**Sample Input and Output 2:**

Enter the book details(code,title,author,price)  
**COM1F,Code Complete,McConnell,1500**  
InvalidBookException: Invalid book code

Top of Form

Bottom of Form

public class Book {

private String code;

private String title;

private String author;

private double price;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

//fill your code here

public String toString() {

//fill your code here

}

}

**public** **class** BookBO {

**public** Book createBook(String details) {

Book b;

b=**new** Book();

**char** arr[]=details.toCharArray();

**return** **null**;

//fill your code here

}

}

public class InvalidBookException extends Exception{

//fill your code here

}

public class Main {

public static void main(String[] args) throws Exception{

//fill your code here

}

}