QuestionBank Assignment3.zip

Java Question Bank

Question 1.)

You are developing a Java program for a simple library management system using an ArrayList to manage books in the library. Each book should have attributes including the title, author, and ISBN number. Implement a Book class with methods to perform the following actions:

1. Allow users to add a new book to the library.
2. Allow users to remove a book from the library by providing its ISBN number.
3. Enable users to search for a book by title and display its details.
4. Display the list of all books currently in the library.

```
    Library Menu
1. Add Book
2. Remove Book by ISBN
3. Search Book by Title
4. Display All Books
5. Exit
Your choice = 1
Enter Title : Java
Enter Author: James Gosling
Enter ISBN  : 1001
Book added successfully!
    Library Menu
1. Add Book
2. Remove Book by ISBN
3. Search Book by Title
4. Display All Books
5. Exit
Your choice = 4
     Book List
Title  = Java
Author = James Gosling
ISBN   = 1001
Exit
    Library Menu
1. Add Book
2. Remove Book by ISBN
3. Search Book by Title
4. Display All Books
5. Exit
Your choice = 1
Enter Title : Python
Enter Author: van
Enter ISBN  : 1002
Book added successfully!
    Library Menu
1. Add Book
```

```java
import java.util.*;
class Book {
    private String title;
    private String author;
    private String isbn;
    public Book(String title, String author, String isbn) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
    }
    public String getTitle(){
        return title;
        }
    public String getAuthor(){
        return author;
        }
    public String getIsbn(){
        return isbn;
        }
    public void display(){
        System.out.println("Title  = " + title);
        System.out.println("Author = " + author);
        System.out.println("ISBN   = " + isbn);
    }
}

public class LibraryQue1{
    private static ArrayList<Book> library = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        int choice;
        do {
            System.out.println("    Library Menu ");
            System.out.println("1. Add Book");
            System.out.println("2. Remove Book by ISBN");
            System.out.println("3. Search Book by Title");
            System.out.println("4. Display All Books");
            System.out.println("5. Exit");
```

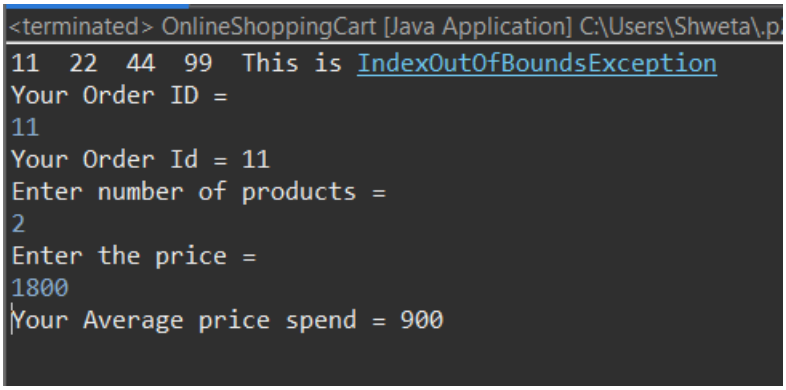
```
            System.out.print("Your choice = ");
            choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
            case 1:{
                addBook();
                break;
            }
            case 2:{
                removeBook();
                break;
            }
            case 3:{
                searchBook();
            }
            case 4:{
                displayBooks();
            }
            case 5:{
                System.out.println("Exit");
                break;
            }
            default:{
                System.out.println("Invalid choice");
            }
            }
        }
    while (choice != 5);


ivate static void addBook() {
    System.out.print("Enter Title : ");
    String title = scanner.nextLine();
    System.out.print("Enter Author: ");
    String author = scanner.nextLine();
    System.out.print("Enter ISBN  : ");
    String isbn = scanner.nextLine();
```

```
  String isbn = scanner.nextLine();
  boolean removed = false;
  Iterator<Book> iterator = library.iterator();

  while (iterator.hasNext()) {
      Book book = iterator.next();
      if (book.getIsbn().equals(isbn)) {
          iterator.remove();
          removed = true;
          break;
      }
  }
  if (removed) {
      System.out.println("Book removed successfully.");
  } else {
      System.out.println("Book not found.");
  }

ivate static void searchBook() {
  System.out.print("Enter title to search: ");
  String title = scanner.nextLine();
  boolean found = false;
  for (Book book : library) {
      if (book.getTitle().equalsIgnoreCase(title)) {
          System.out.println("Book Found!");
          book.display();
          found = true;
          break;
  }
  }}
  private static void displayBooks() {
      if (library.isEmpty()) {
          System.out.println("Library is empty");
      } else {
          System.out.println("     Book List");
          for (Book book : library) {
              book.display();
```

Question 2.)

You are developing a Java program to manage an online shopping cart. Implement code to handle the following built-in exceptions:

1. ArrayIndexOutOfBoundsException
2. NumberFormatException
3. ArithmeticException


<terminated> OnlineShoppingCart [Java Application] C:\Users\Shweta\.p
11 22 44 99 This is IndexOutOfBoundsException
Your Order ID =
11
Your Order Id = 11
Enter number of products =
2
Enter the price =
1800
Your Average price spend = 900

```java
import java.util.Scanner;

public class OnlineShoppingCart{

    public static void main(String[] args) {
        int arr[] = {11, 22, 44, 99};
        try {
            for (int i = 0; i <= arr.length; i++) {
                System.out.print(arr[i] + "  ");
            }
        } catch (IndexOutOfBoundsException ie) {
            System.out.println("This is IndexOutOfBoundsException");
        }

        try {
            Scanner scan = new Scanner(System.in);
            System.out.println("Your Order ID = ");
            String str = scan.nextLine();
            int number = Integer.parseInt(str);
            System.out.println("Your Order Id = " + number);
        } catch (NumberFormatException ne) {
            System.out.println("This is NumberFormatException, ID should not be in Alpha numeric"
        }

        try {
            Scanner scan = new Scanner(System.in);
            System.out.println("Enter number of products = ");
            int p = scan.nextInt();
            System.out.println("Enter the price = ");
            int totalp = scan.nextInt();

            int avg = totalp / p;
            System.out.println("Your Average price spend = " + avg);

        } catch (ArithmeticException ae) {
            System.out.println("This is ArithmeticException");
        }
    }
}
```

Question 3.)

You are developing a Java application for managing vehicles in a rental service. Create a class named Vehicle with attributes for make, model, and year. Implement a constructor that initializes these fields. Next, design a subclass named Car with an additional attribute for numberOfDoors. Provide two constructors for the Car class: one that accepts all fields including make, model, year, and numberOfDoors, and another that accepts only make and model, chaining to the superclass constructor. Validate the input in each constructor and provide appropriate getter and setter methods to access and modify the attributes of the Vehicle class. Ensure proper usage of inheritance principles.

```
PS C:\Users\Shweta\Documents\Java Assignment3> java VehicleQue3
Vehicle Rental Menu
1. Add Car
2. Add Car (Make & Model Only)
3. Display All Cars
4. Exit
Enter your choice = 1
Enter Make = BMW
Enter Model = Q9
Enter Year = 2007
Enter Number of Doors = 4
Car added successfully!

Vehicle Rental Menu
1. Add Car
2. Add Car (Make & Model Only)
3. Display All Cars
4. Exit
Enter your choice = 2
Enter Make = Audi
Enter Model = Q7
Car added with default year (2023) and 4 doors

Vehicle Rental Menu
1. Add Car
2. Add Car (Make & Model Only)
3. Display All Cars
4. Exit
Enter your choice = 3
Car List
Make = BMW
Model= Q9
Year = 2007
numberOfDoors = 4
Make = Audi
Model= Q7
Year = 2023
numberOfDoors = 4

Vehicle Rental Menu
1. Add Car
2. Add Car (Make & Model Only)
3. Display All Cars
4. Exit
Enter your choice = 4
Exit
```

```java
import java.util.*;

class Vehicle {

    private String make, model;
    private int year;

    public Vehicle(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    public String getMake() {
        return make;
    }

    public void setMake(String make) {
        this.make = make;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public void displayInfo() {
        System.out.println("Make = " + make);
        System.out.println("Model= " + model);
        System.out.println("Year = " + year);
    }
}

class Car extends Vehicle {

    private int numberOfDoors;

    public Car(String make, String model, int year, int numberOfDoors) {
        super(make, model, year);
        this.numberOfDoors = numberOfDoors;
    }

    public Car(String make, String model) {
        super(make, model, 2023);
        this.numberOfDoors = 4;
    }

    public int getNumberOfDoors() {
        return numberOfDoors;
    }

    public void setNumberOfDoors(int numberOfDoors) {
        this.numberOfDoors = numberOfDoors;
    }
```

```java
public class VehicleQue3 {

    private static final ArrayList<Car> carList = new ArrayList<>();
    private static final Scanner scanner = new Scanner(System.in);

    Run main | Debug main
    public static void main(String[] args) {
        int choice;
        do {
            System.out.println("Vehicle Rental Menu");
            System.out.println("1. Add Car");
            System.out.println("2. Add Car (Make & Model Only)");
            System.out.println("3. Display All Cars");
            System.out.println("4. Exit");
            System.out.print("Enter your choice = ");
            choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1:
                    addFullCar();
                    break;
                case 2:
                    addSimpleCar();
                    break;
                case 3:
                    displayCars();
                    break;
                case 4:
                    System.out.println("Exit");
                    break;
                default:
                    System.out.println("Invalid choice");
            }
            scanner.nextLine();
            choice = 0;

        } while (choice != 4);
        scanner.close();
    }

    private static void addFullCar() {
        System.out.print("Enter Make = ");
        String make = scanner.nextLine();
        System.out.print("Enter Model = ");
        String model = scanner.nextLine();
        System.out.print("Enter Year = ");
        int year = scanner.nextInt();
        System.out.print("Enter Number of Doors = ");
        int doors = scanner.nextInt();
        scanner.nextLine();
        Car car = new Car(make, model, year, doors);
        carList.add(car);
        System.out.println("Car added successfully!");
    }

    private static void addSimpleCar() {
        System.out.print("Enter Make = ");
        String make = scanner.nextLine();
        System.out.print("Enter Model = ");
        String model = scanner.nextLine();
        Car car = new Car(make, model);
        carList.add(car);
        System.out.println("Car added with default year (2023) and 4 doors");
    }

    private static void displayCars() {
```

Question 4.)

Create a Java program to manage an employee database for a company using an ArrayList. Each employee

should have attributes such as employee name, employee ID, and department. Implement an Employee class with methods to perform the following operations:

1. Add a new employee to the database.
2. Update an employee's department using their employee ID.
3. Remove an employee from the database using their employee ID.
4. Display the list of all employees along with their details.

```
Employee Menu
1. Add Employee
2. Update Department by Employee ID
3. Remove Employee by Employee ID
4. Display All Employees
5. Exit
Your choice = 1
Enter Employee Name = PQR
Enter Employee ID = 102
Enter Department = HR
Employee added successfully!

Employee Menu
1. Add Employee
2. Update Department by Employee ID
3. Remove Employee by Employee ID
4. Display All Employees
5. Exit
Your choice = 4
Employee List
Emp Name    = ABC
Emp ID      = 101
Department = CSE
Emp Name    = PQR
Emp ID      = 102
Department = HR
Exit

Employee Menu
1. Add Employee
2. Update Department by Employee ID
3. Remove Employee by Employee ID
4. Display All Employees
5. Exit
Your choice = 2
Enter Employee ID to update = 101
```

```java
import java.util.*;
class Employee {
    private String empname;
    private int empID;
    private String department;
    public Employee(String empname, int empID, String department) {
        this.empname = empname;
        this.empID = empID;
        this.department = department;
    }

    public String getEmpname() { return empname; }
    public int getEmpID() { return empID; }
    public String getDepartment() { return department; }

    public void setDepartment(String department) {
    }

    public void display() {
        System.out.println("Emp Name    = " + empname);
        System.out.println("Emp ID      = " + empID);
        System.out.println("Department = " + department);
    }
}

public class EmployeeQue4 {
    private static ArrayList<Employee> employeeList = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int choice;
        do {
            System.out.println("\nEmployee Menu");
            System.out.println("1. Add Employee");
            System.out.println("2. Update Department by Employee ID");
            System.out.println("3. Remove Employee by Employee ID");
            System.out.println("4. Display All Employees");
```

```
            choice = scanner.nextInt();
           switch (choice) {
        case 1:{
            addEmployee();
            break;
        }
        case 2:{
            updateEmployee();
            break;
        }
        case 3:{
            removeEmployee();
        }
        case 4:{
            displayEmp();
        }
        case 5:{
            System.out.println("Exit");
            break;
        }
        default:{
            System.out.println("Invalid choice");
        }
        }
    }
while (choice != 5);
}
    private static void addEmployee() {
        System.out.print("Enter Employee Name = ");
        String name = scanner.nextLine();
        System.out.print("Enter Employee ID = ");
        int empID = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Department = ");
        String dept = scanner.nextLine();
        employeeList.add(new Employee(name, empID, dept));
        System.out.println("Employee added successfully!");
```

```
}
private static void removeEmployee() {
    System.out.print("Enter Employee ID to remove = ");
    int empID = scanner.nextInt();
    boolean removed = false;
    Iterator<Employee> iterator = employeeList.iterator();
    while (iterator.hasNext()) {
        Employee emp = iterator.next();
        if (emp.getEmpID()==(empID)) {

            iterator.remove();
            removed = true;
            break;
        }
    }
    if (removed) {
        System.out.println("Employee removed successfully");
    } else {
        System.out.println("Employee not found");
    }
}

private static void displayEmp() {
    if (employeeList.isEmpty()) {
        System.out.println("No employees in the database.");
    } else {
        System.out.println("Employee List");
        for (Employee emp : employeeList) {
            emp.display();
        }
    }
}
```

Question 5.)

You are developing a Java program to manage user inputs in a ticket booking system. Implement code to handle the following built-in exceptions:

1. InputMismatchException3
2. IllegalArgumentException
3. IndexOutOfBoundsException

```java
import java.util.*;
class TicketBookingQue5{
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try{
            String[] seats = {"S1", "S2", "S3", "S4"};

            System.out.print("Enter number of ticket =");
            int tickets = sc.nextInt();

            if (tickets <= 0) {
                throw new IllegalArgumentException("Number of ticket must be positive");
            }
            System.out.print("Choose seat between 0-3 = ");
            int index = sc.nextInt();

            System.out.println("You booked seat = " + seats[index]);

        }
        catch(InputMismatchException e){
            System.out.println("Error: Invalid input type. Please enter a number.");
        }
        catch(IllegalArgumentException e){
            System.out.println("Error: " + e.getMessage());
        }
        catch(IndexOutOfBoundsException e){
            System.out.println("Error: Invalid seat index. Please choose within range.");
        }
        finally{
            System.out.println("Booking process finished.");
            sc.close();
        }
```

```
Enter number of ticket =1
Choose seat between 0-3 = 1
You booked seat = S2
Booking process finished.
```

Question 6.)

You are creating a Java program to represent different types of employees in a company. Implement a class named Employee with attributes for name, employeeID, and department. Provide a constructor to initialize these fields. Next, create a subclass named Manager with an additional attribute for numberOfTeams managed. The Manager class should have two constructors: one that accepts all fields, and another that accepts only name and employeeID, chaining to the superclass constructor. Include methods to validate input data and implement getter and setter methods for the attributes in the Employee class. Demonstrate the inheritance hierarchy by creating instances of both classes and displaying their details.

```
Name = A
empId = 9
Department = CSE
Name = B
empId = 7
Department = null
numOfTManaged = 3
```

```java
class Employeee{
    String name;
    int empId;
    String Department;

    Employeee(String name,int empId,String Department){
        this.name=name;
        this.empId=empId;
        this.Department=Department;
    }
    Employeee(String name,int empId){
        this.name=name;
        this.empId=empId;
    }
    void setName(String name){
        this.name=name;
    }
    void setEmpId(int empId){
        this.empId=empId;
    }
    void setDepartment(String Department){
        this.Department=Department;
    }
    String getName(String name){
        return name;
    }
    int getEmpId(int empId){
        return empId;
    }
    String getDepartment(String Department){
        return Department;
    }
    void display1()
    {
        System.out.println("Name = "+name);
        System.out.println("empId = "+empId);
        System.out.println("Department = "+Department);
    }

}
class Manager extends Employeee{
    int numOfTManaged;
```

```java
class Manager extends Employeee{
    int numOfTManaged;
    Manager(String name,int empId){
        super(name,empId);
    }
    Manager(String name,int empId,String Department,int numOfTManaged){
        super(name,empId,Department);
        this.numOfTManaged=numOfTManaged;
    }

    void setnumOfTManaged(int numOfTManaged){
        this.numOfTManaged=numOfTManaged;
    }
    int getName(int numOfTManaged){
        return numOfTManaged;
    }
    void display()
    {
        System.out.println("Name = "+name);
        System.out.println("empId = "+empId);
        System.out.println("Department = "+Department);
        System.out.println("numOfTManaged = "+numOfTManaged);
    }
}

public class CompanyQue6{
    public static void main(String[] args) {
        Employeee e = new Employeee("A",9,"CSE");
        Manager m =new Manager("B",7);
        m.setnumOfTManaged(3);
        e.display1();
        m.display();
    }
}
```

Question 7.)

Develop a Java program for a simple banking system using an ArrayList to manage bank accounts. Each bank account should have attributes including the name of the account holder, account number, and initial balance. Implement the BankAccount class with methods to perform the following actions: (Using array list)

- Allow users to deposit money into their account.
- Allow users to withdraw money from their account if they have sufficient balance.
- Display the current balance of the account.
- Enable users to transfer money from one account to another, provided they have sufficient balance.

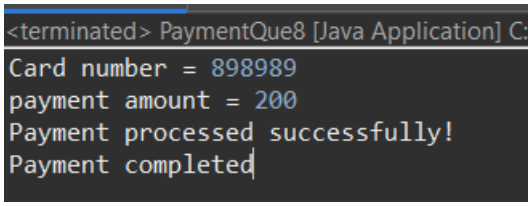
```
Enter your choice = 4
Name : Aaa  Account no : 100001  Balance : 3000
Name : Bbb  Account no : 100002  Balance : 2000
1. Add Account to Database
2. Deposit Money
3. Withdraw Money
4. Display Account Details
5. Transfer Money
6. Exit
Enter your choice = 2
Enter Account number for deposit = 100002
Enter Deposit amount = 300
Money Deposited...
1. Add Account to Database
2. Deposit Money
3. Withdraw Money
4. Display Account Details
5. Transfer Money
6. Exit
Enter your choice = 5
Enter Source Account number = 100001
Enter Destination Account number = 100002
Enter Transfer amount = 200
Transfer successful
1. Add Account to Database
2. Deposit Money
3. Withdraw Money
4. Display Account Details
5. Transfer Money
6. Exit
Enter your choice = 6
Exit
```

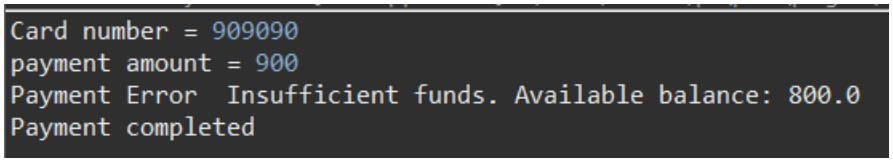
```java
import java.util.*;
class Bank {
    private String name;
    private String accnumber;
    private int balance;

    public Bank(String name, String accnumber, int balance) {
        this.name = name;
        this.accnumber = accnumber;
        this.balance = balance;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setAccnumber(String accnumber) {
        this.accnumber = accnumber;
    }

    public void setBalance(int balance) {
        this.balance = balance;
    }

    public String getName() {
        return name;
    }

    public String getAccnumber() {
        return accnumber;
    }

    public int getBalance() {
        return balance;
    }

    public void display() {
        System.out.println("Name : " + name + "  Account no : " + accnumber + "  Balance : " + balance)
    }
}

public class BankAccountQue7{
```

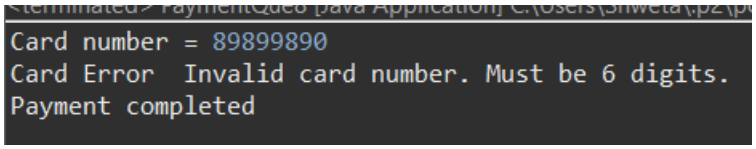
Question 8.)

You're developing error handling for a Java program that manages payments in an e-commerce application. Write code to handle the following three, specific payment gateway errors:

- Timeout error: Implement exception handling to catch situations where the payment gateway response times out.
- Invalid card details: Handle exceptions arising from attempts to process payments with invalid card information.
- Insufficient funds: Implement exception handling to manage cases where users attempt to make payments without sufficient funds in their account.

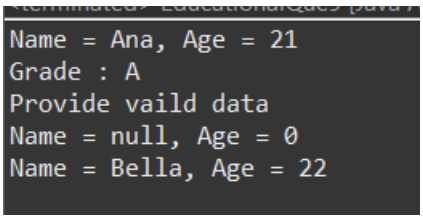






Question 9.)

You're developing a Java program to manage individuals within an educational institution. Create a class named "Person" with attributes for the name and age of an individual. Implement a constructor that accepts both fields as arguments. Next, design a subclass named "Student" with an additional attribute for the grade level. Provide two constructors for the Student class: one that accepts all fields including name, age, and grade level, and another that accepts only the name and age, chaining to the superclass constructor. Ensure that your program demonstrates proper inheritance principles, and validate the constructors to ensure valid data is provided during object instantiation,also implement appropriate getter and setter methods to access and modify the attributes of Person class.



```java
class Person{
    String name;
    int age;

    Person(String name,int age){
        if(age<=0){
            System.out.println("Provide vaild data");
        }
        else{
            this.name=name;
            this.age=age;
        }
    }
    void setname(String name){
        this.name=name;
    }
    void setage(int age){
        if(age<0){
            System.out.println("Provide vaild data");
        }
        else{
            this.age=age;
        }
    }

    String getname(){
        return name;
    }
    int getage(){
        return age;
    }

    void display(){
        System.out.println("Name = "+name+", Age = "+age);
    }
}
class Student extends Person{
    String grade;
```

```
    void display(){
        System.out.println("Name = "+name+", Age = "+age);
    }

lass Student extends Person{
    String grade;

    Student(String name,int age,String grade){
        super(name,age);
        this.grade=grade;
    }
    Student(String name,int age){
        super(name,age);
    }

    void setgrade(String grade){
        this.grade=grade;
    }
    String getgrade(){
        return grade;
    }

    void displayAll(){
        super.display();
        System.out.println("Grade : "+grade);
    }

ublic class EducationalQue9{
    public static void main(String[] args) {
        Student s=new Student("Ana",21,"A");
        s.displayAll();
        Student s1=new Student("Bella",0);
        s1.display();
        s1.setage(22);
        s1.setname("Bella");
        s1.display();
```

sole ×

Question 10)

Create a class Student with:

- int studentId, String name, double grade.
- A constructor to initialize these fields.
- Methods:
    - updateGrade(double newGrade): Updates the grade, but should not accept negative values (handle using exception handling).

○ display(): Prints student details.

```
Student Id = 101, Name = Anaa, Grade = 89.9
Enter valid Grade
Updated Grade
Student Id = 101, Name = Anaa, Grade = 98.0
```

```java
class Studentt{
    int studentId;
    String name;
    double grade;

    Studentt(int studentId, String name, double grade){
        this.studentId=studentId;
        this.name=name;
        this.grade=grade;
    }

    void updateGrade(double newGrade){
        try {
            if(grade<=0){
                throw new IllegalArgumentException("Grade must be positive");
            }

        } catch (IllegalArgumentException  e) {
            System.out.println("Enter valid Grade");
        }
        this.grade=newGrade;
    }
    void setStudentId(int studentId){
        this.studentId=studentId;
    }
    void setname(String name){
        this.name=name;
    }

    int getstudentId(){
        return studentId;
    }
    String getname(){
        return name;
    }
    double getgrade(){
        return grade;
```

```
void updateGrade(double newGrade){
    try {
        if(grade<=0){
            throw new IllegalArgumentException("Grade must be positive");
        }

    } catch (IllegalArgumentException  e) {
        System.out.println("Enter valid Grade");
    }
    this.grade=newGrade;
}
void setStudentId(int studentId){
    this.studentId=studentId;
}
void setname(String name){
    this.name=name;
}

int getstudentId(){
    return studentId;
}
String getname(){
    return name;
}
double getgrade(){
    return grade;
}
void display(){
    System.out.println("Student Id = "+studentId+", Name = "+name+", Grade = "+grade)
}


lic class StudentDetailsQue11{
public static void main(String[] args) {
    Studentt std = new Studentt(101,"Anaa",89.9);
    std.display();
    std.updateGrade(0);
```

Question 11)

Write a Java program that reads a string from the user and attempts to convert it to an integer using Integer.parseInt(). If the input is not a valid integer, handle the NumberFormatException. Additionally, handle NullPointerException if the input is null. Use a finally block to print "Conversion attempt completed."

```
Enter you account number = a
String can not be converted to Integer
Conversion attempt completed, Account Number = 0
```

```
Enter you account number = 1008999
Conversion attempt completed, Account Number = 1008999
```

```java
import java.util.Scanner;
public class NumberExceptionQue11 {
        public static void main(String[] args) {
            String accountno;
            Scanner scan=new Scanner(System.in);
            System.out.print("Enter you account number = ");
            accountno=scan.nextLine();
            int accnumber = 0;

            try {
                accnumber=Integer.parseInt(accountno);

            } catch (NumberFormatException ne) {
                System.out.println("String can not be converted to Integer");
            }
            catch(NullPointerException npe){
                System.out.println("String can not be Null");
            }
            finally{
                System.out.println("Conversion attempt completed, Account Number = "+accnumber);
            }
            scan.close();
        }
    }
```

Question 12)

Patient Management System

You are managing a patient database for a hospital. Each patient has a unique patient ID, a name, a diagnosis, and the number of days admitted. You need to implement a solution using appropriate Java collection classes to efficiently perform the following operations: a. Add a new patient to the database.

b. Remove a patient from the database.

c. Find all patients with a specific diagnosis.

d. Find all patients admitted for more than a given number of days.

```
Enter your choice = 1
Enter Patient Id = 22
Enter Patient Name = Pqr
Enter Patient Diagonsis = can
Enter Number of days Admitted = 22
1. Add Patient Details
2. Remove Patient
3. Find all patients with a specific diagnosis
4. Find all patients admitted for more than a 5 days
5. Display all Patient
6. Exit
Enter your choice = 5
Displaying all patient
Patient Id : 11 Name : Abc Diagnosis : Cancer Admited Days : 24
Patient Id : 22 Name : Pqr Diagnosis : can Admited Days : 22
1. Add Patient Details
2. Remove Patient
3. Find all patients with a specific diagnosis
4. Find all patients admitted for more than a 5 days
5. Display all Patient
6. Exit
Enter your choice = 3
Search Patient with special Diagonsis = can
Patient Id : 22 Name : Pqr Diagnosis : can Admited Days : 22
1. Add Patient Details
2. Remove Patient
3. Find all patients with a specific diagnosis
4. Find all patients admitted for more than a 5 days
5. Display all Patient
6. Exit
Enter your choice = 4
Number of Patient admitted more than 5 =
Patient Id : 11 Name : Abc Diagnosis : Cancer Admited Days : 24
Patient Id : 22 Name : Pqr Diagnosis : can Admited Days : 22
1. Add Patient Details
2. Remove Patient
```