Experiment5.1

Student Name: Amit Kumar Raj	**UID: 22BCS13131**
Branch:CSE	**Section:22BCS-IOT-641-A**
Semester:6th	**DOP:14-02-2025**
Subject:PBLJ	**SubjectCode:22CSH-359**

Aim: Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing. Include methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

Objective: Demonstrate **autoboxing** and **unboxing** in Java by converting string numbersinto Integer objects, storing them in a list, and computing their sum.

Algorithm:

Step1: Initializethe Program

1. Startthe program.
2. ImportArrayListandList classes.
3. DefinetheAutoboxingExample class.

Step2:ConvertString ArraytoIntegerList

1. DefinethemethodparseStringArrayToIntegers(String[]strings).
2. Createan emptyArrayList<Integer>.
3. Iteratethroughthestringarray:
   - ConverteachstringtoanIntegerusingInteger.parseInt(str).
   - Addtheintegerto thelist(**autoboxing**happens here).
4. Returnthe list of integers.

Step3:CalculatetheSum of Integers

1. DefinethemethodcalculateSum(List<Integer>numbers).
2. Initializeavariable sumto0.
3. Iteratethroughthe list:
   - Extracteachinteger(**unboxing** happenshere).
   - Add it to sum.
4. Returnthe total sum.

Step4:ExecuteMain Function

1. Definemain(String[]args).
2. Createastring arraywithnumeric values.
3. CallparseStringArrayToIntegers() toconvertitintoalistofintegers.
4. CallcalculateSum()to computethe sum.
5. Print theresult.

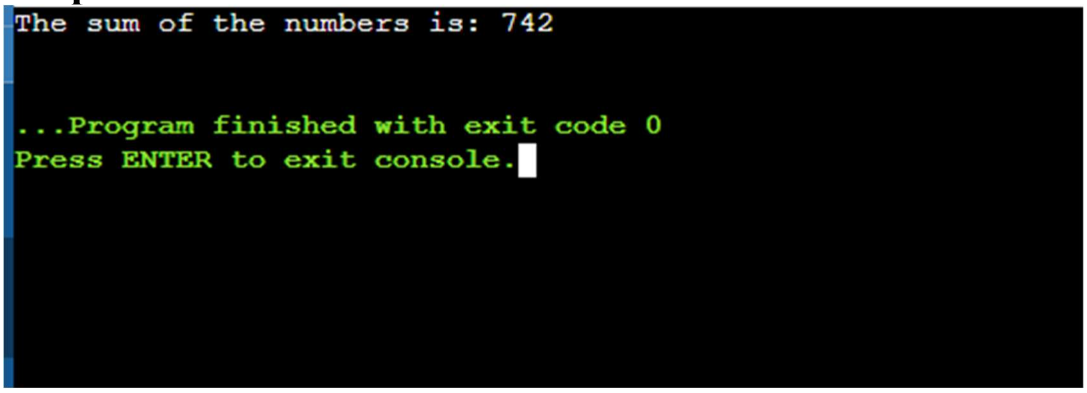
Step5: Terminatethe Program

1. End the execution.

Code:

```
importjava.util.ArrayList;
import java.util.List;

public class AutoboxingExample
  {publicstaticvoidmain(String[]args){
    String[] numberStrings = {"90", "25", "30", "57", "540"}

    List<Integer>numbers=parseStringArrayToIntegers(numberStrings);

    int sum = calculateSum(numbers);

    System.out.println("Thesumof thenumbersis: "+ sum);
  }

  publicstaticList<Integer>parseStringArrayToIntegers(String[]strings){
    List<Integer> integerList = new ArrayList<>();
    for (String str : strings) {
      integerList.add(Integer.parseInt(str));
    }
    return integerList;
  }

  publicstaticintcalculateSum(List<Integer>numbers){ int
    sum = 0;
    for(Integernum:numbers){ sum
      += num;
    }
    returnsum;
  }
}
```

Output:



```
The sum of the numbers is: 742


...Program finished with exit code 0
Press ENTER to exit console.
```

LearningOutcomes:

- Understand the concept of **autoboxing and unboxing** in Java and how primitive types are automatically converted to their wrapper classes and vice versa.
- Learnhowto **convertstringvaluesintoIntegerobjects** using Integer.parseInt()andstore them in a list.
- Gainexperiencein**workingwithArrayLists**tostoreandmanipulateacollectionof numbers dynamically.
- Developproficiencyin**iteratingthroughcollections**andperformingarithmetic operations like summation.

Experiment5.2

1. **Aim:**CreateaJavaprogramtoserializeanddeserializeaStudentobject. The program should:

- SerializeaStudentobject(containing id,name,andGPA)and saveittoafile.
- Deserializetheobjectfrom thefileand displaythe student details.
- HandleFileNotFoundException,IOException,andClassNotFoundExceptionusingexception handling.

2. **Objective:** The objective is to serialize and deserialize a Student object, store and retrieve its id, name, and GPA froma file, and handleexceptions like FileNotFoundException, IOException, and ClassNotFoundException.

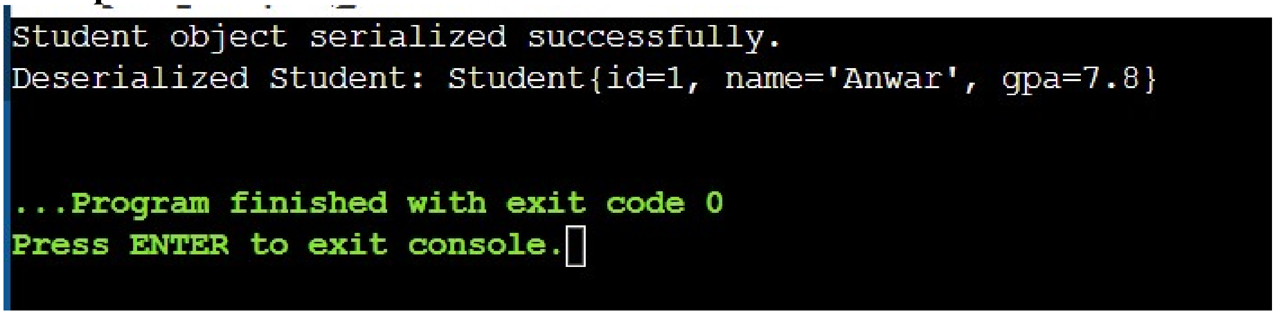
3. Algorithm:

```
Step1:InitializetheProgram
    1. Startthe program.
    2. Importthenecessaryclasses(java.io.*).
    3. DefineaStudent classimplementingSerializable.
    4. Declareattributes:
        o id (int)
        o name (String)
        o gpa(double)
    5. Defineaconstructorto initializeStudent objects.
    6. OverridetoString()todisplaystudentdetails.
Step 2: Define the Serialization Method
    1. CreateserializeStudent(Student student).
    2. Useatry-with-resourcesblocktocreateanObjectOutputStream:
        o Opena FileOutputStream towriteto student.ser.
        o Writethe Studentobjectto thefileusingwriteObject().
    3. Handleexceptions:
        o FileNotFoundException →Printerror message.
        o IOException→Printerrormessage.
    4. Printasuccessmessageifserializationissuccessful.
Step 3: Define the Deserialization Method
    1. CreatedeserializeStudent().
    2. Useatry-with-resourcesblocktocreateanObjectInputStream:
        o OpenaFileInputStreamtoreadstudent.ser.
        o ReadtheStudentobjectusingreadObject().
    3. Handleexceptions:
        o FileNotFoundException →Printerror message.
        o IOException→Printerrormessage.
        o ClassNotFoundException→Printerrormessage.
    4. Printthedeserializedstudentdetails.
Step 4: Execute Main Function
    1. Definemain(String[]args).
    2. CreateaStudent objectwithsampledata.
    3. CallserializeStudent()tosavetheobject.
    4. CalldeserializeStudent()toreadanddisplaytheobject.
Step 5: Terminate the Program
    1. End execution.
```

4. ImplementationCode:

```
import java.io.*;

classStudentimplementsSerializable{
   privatestaticfinallongserialVersionUID=1L; private
   int id;
   privateStringname;
   private double gpa;

   publicStudent(intid,Stringname,doublegpa){ this.id =
      id;
      this.name=name;
      this.gpa = gpa;
   }

   @Override
   publicStringtoString(){
      return"Student{id=" +id+ ", name='" +name +"', gpa="+ gpa+"}";
   }
}

publicclassStudentSerialization{
   privatestaticfinalStringFILE_NAME="student.ser";

   publicstatic voidmain(String[]args){
      Studentstudent=newStudent(1,"Anwar",7.8);
      serializeStudent(student);
      deserializeStudent();
   }

   publicstaticvoidserializeStudent(Studentstudent){
      try(ObjectOutputStreamoos=newObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
         oos.writeObject(student);
         System.out.println("Studentobjectserializedsuccessfully.");
      } catch (FileNotFoundException e) {
         System.err.println("Filenotfound:"+e.getMessage());
      }catch(IOExceptione){
         System.err.println("IOExceptionoccurred:"+e.getMessage());
      }
   }

   publicstaticvoiddeserializeStudent(){
      try(ObjectInputStreamois=newObjectInputStream(new FileInputStream(FILE_NAME)))
{
         Student student = (Student) ois.readObject();
         System.out.println("DeserializedStudent:"+student);
      } catch (FileNotFoundException e) {
         System.err.println("Filenotfound:"+e.getMessage());
      }catch(IOExceptione){
         System.err.println("IOExceptionoccurred:"+e.getMessage());
      }catch(ClassNotFoundExceptione){
            System.err.println("Classnotfound:"+e.getMessage());
        }
    }
}
```

5. Output



```
Student object serialized successfully.
Deserialized Student: Student{id=1, name='Anwar', gpa=7.8}


...Program finished with exit code 0
Press ENTER to exit console.
```

6. LearningOutcomes:

- UnderstandobjectserializationanddeserializationinJava.
- LearnhowtouseObjectOutputStream andObjectInputStreamforfileoperations.
- Implement exception handling for FileNotFoundException, IOException, and ClassNotFoundException.
- Gainhands-onexperienceinstoringandretrievingobjects froma file.
- Developskillsin datapersistenceandfilemanagementusing Java.

Experiment5.3

1. **Aim:**Create amenu-basedJavaapplication withthefollowing options.
   1. AddanEmployee
   2. DisplayAll
   3. Exit If option 1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected, the application should display all the employee details. If option 3 is selected the application should exit.
2. **Objective:**The objective is to develop a menu-based Java application that allows users to **add employee details**, **store them in a file**, and **display all stored employee records**, with an option to exit the program.

3. **Algorithm:**

Step1: Initializethe Program

1. Startthe program.
2. Importjava.util.*andjava.util.concurrent.*forthreadhandling.
3. DefineaclassTicketBookingSystem with:
   - AList<Boolean>representingseatavailability(trueforavailable,falsefor booked).
   - AsynchronizedmethodbookSeat(intseatNumber,StringpassengerName)to ensure thread safety.

Step2:Implement SeatBookingLogic

1. DefinebookSeat(intseatNumber,String passengerName):
   - Iftheseatisavailable(true),markitasbooked(false).
   - Printconfirmation:"SeatX bookedsuccessfullyby Y".
   - Ifalreadybooked,print:"SeatXisalready booked."

Step3:DefineBookingThreads

1. CreateaclassPassengerThread extendingThread:
   - Storepassengername, seatnumber,and bookingsystem reference.
   - Implementrun()method tocall bookSeat().

Step4:AssignThreadPriorities

1. CreateVIPandRegularpassengerthreads.
2. SethigherpriorityforVIPpassengersusing setPriority(Thread.MAX_PRIORITY).
3. Setdefaultpriority forregular passengers.

Step5:Handle UserInput &SimulateBooking

1. Inmain(),createaninstanceof TicketBookingSystem.
2. Acceptnumber ofseats and bookingsfrom theuser.
3. CreatemultiplePassengerThreadinstancesforVIPandregularpassengers.
4. Startallthreadsusingstart().

Step6:Synchronization&PreventingDoubleBooking

1. UsethesynchronizedkeywordinbookSeat()toensureonlyonethreadaccessesitata time.
2. EnsurethreadexecutionorderbyassigninghigherprioritytoVIPthreads.

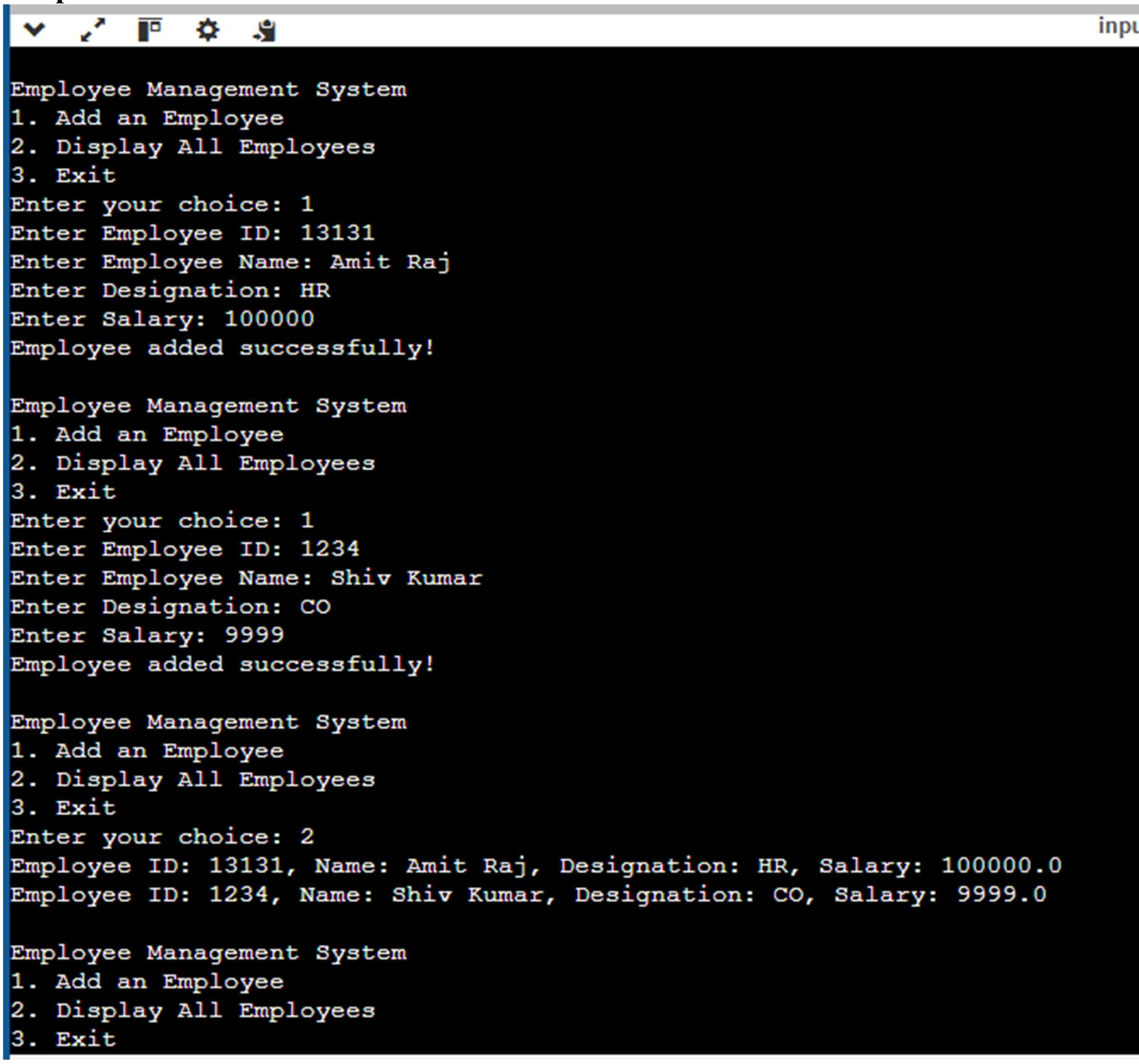
Step7:DisplayFinalBooking Status

1. Afterallthreads finishexecution, displaythelist ofbooked seats.
2. Endtheprogramwitha message:"Allbookingscompleted successfully."

4. ImplementationCode:

```
importjava.io.*;
import java.util.*;

classEmployeeimplementsSerializable{
    privatestaticfinallongserialVersionUID=1L;
    private int id;
    private String name;
    privateStringdesignation;
    private double salary;

    publicEmployee(intid,Stringname,Stringdesignation,doublesalary){ this.id =
            id;
            this.name = name;
            this.designation=designation;
            this.salary = salary;
    }

    @Override
    publicString toString(){
            return"EmployeeID:"+id+",Name:"+name+",Designation:"+designation
+", Salary:" +salary;
    }
}

publicclassEmployeeManagementSystem{
    private static final String FILE_NAME = "employees.ser";
    privatestaticList<Employee>employees=newArrayList<>();

    publicstaticvoidaddEmployee(){
            Scannerscanner=newScanner(System.in);
            System.out.print("Enter Employee ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.print("EnterEmployeeName:");
            String name = scanner.nextLine();
            System.out.print("Enter Designation: ");
            String designation = scanner.nextLine();
            System.out.print("Enter Salary: ");
            doublesalary= scanner.nextDouble();

            Employeeemployee=newEmployee(id,name,designation,salary); employees.add(employee);
            saveEmployees();
            System.out.println("Employeeadded successfully!");
    }

    publicstaticvoiddisplayAllEmployees(){
            loadEmployees();
            if(employees.isEmpty()){
                    System.out.println("Noemployees found.");
            }else {
                    for(Employeeemployee:employees){
                            System.out.println(employee);
                    }
            }
    }
    privatestaticvoidsaveEmployees(){
            try      (ObjectOutputStream      oos      =      new      ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
                    oos.writeObject(employees);
             }catch(IOExceptione) {
                     System.err.println("Errorsavingemployees:"+ e.getMessage());
            }
    }

    @SuppressWarnings("unchecked")
    privatestaticvoidloadEmployees(){
            try      (ObjectInputStream      ois      =      new      ObjectInputStream(new
FileInputStream(FILE_NAME))) {
                    employees=(List<Employee>) ois.readObject();
            } catch (FileNotFoundException e) {
                    employees=newArrayList<>();
            } catch (IOException | ClassNotFoundException e) {
                    System.err.println("Errorloadingemployees:"+e.getMessage());
            }
    }

    publicstatic voidmain(String[]args){
            Scannerscanner=newScanner(System.in); while
            (true) {
                    System.out.println("\nEmployeeManagementSystem");
                    System.out.println("1. Add an Employee");
                    System.out.println("2. Display All Employees");
                    System.out.println("3. Exit");
                    System.out.print("Enteryourchoice:"); int
                    choice = scanner.nextInt();
                    scanner.nextLine();

                    switch(choice){
                    case 1:
                            addEmployee();
                            break;
                    case 2:
                            displayAllEmployees();
                            break;
                    case 3:
                            System.out.println("Exiting...");
                            return;
                    default:
                            System.out.println("Invalidchoice!Pleasetryagain.");
                    }
            }
    }
}
```

5. **Output:**



```
Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 13131
Enter Employee Name: Amit Raj
Enter Designation: HR
Enter Salary: 100000
Employee added successfully!

Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 1234
Enter Employee Name: Shiv Kumar
Enter Designation: CO
Enter Salary: 9999
Employee added successfully!

Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2
Employee ID: 13131, Name: Amit Raj, Designation: HR, Salary: 100000.0
Employee ID: 1234, Name: Shiv Kumar, Designation: CO, Salary: 9999.0

Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
```

6. **LearningOutcomes:**

- Understandfilehandling andserializationinJavato storeand retrieveobjects persistently.
- Learnhowtoimplementamenu-drivenconsoleapplicationusingloops andconditional statements.
- Gainexperienceinobject-orientedprogramming(OOP)bydefiningandmanaging Employee objects.
- Practice exception handling to manage file-related errors like FileNotFoundException and IOException.
- Developskillsin listmanipulationanduserinputhandlingusingArrayList andScanner.