Q1. Sort a list of students by roll number (ascending) using Comparable.

Create a Student class with fields: rollNo, name, and marks. Implement the Comparable interface to sort students by their roll numbers.

Code: **package** MAp\_collections;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.List;

**class** Student\_1 **implements** Comparable<Student\_1>{

**int** rollno;

String name;

**int** marks;

Student\_1(**int** rollno,String name,**int** marks){

**this**.rollno=rollno;

**this**.name=name;

**this**.marks=marks;

}

**public** **int** compareTo(Student\_1 s) {

**return** rollno-s.rollno;

}

**public** String toString() {

**return** rollno+" "+name;

}

}

**public** **class** Test\_Comparable1 {

**public** **static** **void** main(String[] args) {

List<Student\_1> list=Arrays.*asList*(

**new** Student\_1(9,"Amit",95),

**new** Student\_1(3,"Rahul",89),

**new** Student\_1(2,"Kiran",90)

);

Collections.*sort*(list);//uses comparable

System.***out***.println(list);

}

}

Output; [2 Kiran, 3 Rahul, 9 Amit]

Q2. Create a Product class and sort products by price using Comparable.

Implement Comparable<Product> and sort a list of products using Collections.sort().

Code; **package** MAp\_collections;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.List;

**class** Product **implements** Comparable<Product>{

**int** price;

String productname;

Product(**int** price,String productname){

**this**.price=price;

**this**.productname=productname;

}

**public** **int** compareTo(Product p) {

**return** price-p.price;

}

**public** String toString() {

**return** price+" "+productname;

}

}

**public** **class** Test\_Comparable2 {

**public** **static** **void** main(String[] args) {

List<Product> list=Arrays.*asList*(

**new** Product(100,"Thums up"),

**new** Product(150,"Dairymilk"),

**new** Product(50,"Banana Juice")

);

Collections.*sort*(list);//uses comparable

System.***out***.println(list);

}

}

Output; [50 Banana Juice, 100 Thums up, 150 Dairymilk]

Q3. Create an Employee class and sort by name using Comparable.

Use the compareTo() method to sort alphabetically by employee names.

Code; **package** MAp\_collections;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.List;

**class** Employee **implements** Comparable<Employee>{

**int** id;

String employeename;

Employee(**int** id,String employeename){

**this**.id=id;

**this**.employeename=employeename;

}

**public** **int** compareTo(Employee otheremployee) {

**return** **this**.employeename.compareTo(otheremployee.employeename);

}

**public** String toString() {

**return** id+" "+employeename;

}

}

**public** **class** Test\_Comparable3 {

**public** **static** **void** main(String[] args) {

List<Employee> list=Arrays.*asList*(

**new** Employee(100,"Ravi kumar"),

**new** Employee(150,"Rajesh"),

**new** Employee(500,"Bharghav")

);

Collections.*sort*(list);//uses comparable

System.***out***.println(list);

}

}

Output; [500 Bharghav, 150 Rajesh, 100 Ravi kumar]

Q4. Sort a list of Book objects by bookId in descending order using Comparable.

Hint: Override compareTo() to return the reverse order.

Code; **package** MAp\_collections;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.List;

**class** Book1 **implements** Comparable<Book1>{

**int** id;

String bookname;

Book1(**int** id,String bookname){

**this**.id=id;

**this**.bookname=bookname;

}

**public** **int** compareTo(Book1 other) {

**return** Integer.*compare*(other.id,**this**.id);

}

**public** String toString() {

**return** id+" "+bookname;

}

}

**public** **class** Test\_Comparable4 {

**public** **static** **void** main(String[] args) {

List<Book1> list=Arrays.*asList*(

**new** Book1(100,"Jugle book"),

**new** Book1(150,"Wings of fire"),

**new** Book1(500,"ignited Minds")

);

Collections.*sort*(list);//uses comparable

System.***out***.println(list);

}

}

Output; [500 ignited Minds, 150 Wings of fire, 100 Jugle book]

Q5. Implement a program that sorts a list of custom objects using Comparable, and displays them before and after sorting.

Code; **package** MAp\_collections;

**import** java.util.\*;

**class** Student **implements** Comparable<Student>{

**int** id;

String name;

Student(**int** id,String name){

**this**.id=id;

**this**.name=name;

}

**public** **int** compareTo(Student s) {

**return** id-s.id;

}

**public** String toString() {

**return** id+" "+name;

}

}

**public** **class** TestComparable {

**public** **static** **void** main(String[] args) {

List<Student> list=Arrays.*asList*(

**new** Student(9,"Amit"),

**new** Student(0,"Rahul"),

**new** Student(2,"Kiran")

);

System.***out***.println("Before sorting:");

list.forEach(System.***out***::println);

Collections.*sort*(list);//uses comparable

System.***out***.println("\n After sorting by age:");

System.***out***.println(list);

}

}

Output; Before sorting:

9 Amit

0 Rahul

2 Kiran

After sorting by age:

[0 Rahul, 2 Kiran, 9 Amit]

Q6. Sort a list of students by marks (descending) using Comparator.

Create a Comparator class or use a lambda expression to sort by marks.

Code; **package** MAp\_collections;

**import** java.util.Arrays;

**import** java.util.List;

**class** Student6 **implements** Comparable<Student6>{

String name;

**int** marks;

Student6(String name,**int** marks){

**this**.name=name;

**this**.marks=marks;

}

**public** **int** compareTo(Student6 s) {

**return** marks-s.marks;

}

**public** String toString() {

**return** name+" "+marks;

}

}

**public** **class** Sort\_by\_marks {

**public** **static** **void** main(String[] args) {

List<Student6> list=Arrays.*asList*(

**new** Student6("Amit",70),

**new** Student6("Rahul",90),

**new** Student6("Kiran",95)

);

System.***out***.println("Before sorting:");

list.forEach(System.***out***::println);

list.sort((s1, s2) -> Integer.*compare*(s2.marks, s1.marks));

System.***out***.println("\nAfter sorting by marks (descending):");

list.forEach(System.***out***::println);

}

}

Output; Before sorting:

Amit 70

Rahul 90

Kiran 95

After sorting by marks (descending):

Kiran 95

Rahul 90

Amit 70

Q7. Create multiple sorting strategies for a Product class.

Implement comparators to sort by:

Price ascending

Price descending

Name alphabetically

Code; **package** MAp\_collections;

**import** java.util.\*;

**import** java.lang.\*;

**class** Product4 {

**int** price;

String name;

Product4(**int** price,String name){

**this**.price=price;

**this**.name=name;

}

**public** **int** getPrice() {

**return** price;

}

**public** String toString() {

**return** name +"-Rs. "+price;

}

}

**class** PriceAscComparator **implements** Comparator<Product4>{

**public** **int** compare(Product4 p1,Product4 p2) {

**return** Integer.*compare*(p1.price,p2.price);

}

}

**class** PriceDescComparator **implements** Comparator<Product4>{

**public** **int** compare(Product4 p1,Product4 p2) {

**return** Integer.*compare*(p2.price,p1.price);

}

}

**class** NameComparator **implements** Comparator<Product4> {

**public** **int** compare(Product4 p1, Product4 p2) {

**return** p1.name.compareToIgnoreCase(p2.name);

}

}

**public** **class** multiple\_sorting\_comparators {

**public** **static** **void** main(String[] args) {

List<Product4> products= Arrays.*asList*(

**new** Product4(20,"dairymilk"),

**new** Product4(100,"Thumsup"),

**new** Product4(30,"Lays"));

System.***out***.println("Before sorting:");

products.forEach(System.***out***::println);

products.sort(Comparator.*comparingInt*(p -> p.price));

System.***out***.println("\nAfter sorting by price(descending):");

products.forEach(System.***out***::println);

products.sort(Comparator.*comparingInt*(Product4::getPrice).reversed());

System.***out***.println("\nSorted by price (ascending):");

products.forEach(System.***out***::println);

// Sort by name alphabetically

products.sort(Comparator.*comparing*(p -> p.name.toLowerCase()));

System.***out***.println("\nSorted by name (alphabetically):");

products.forEach(System.***out***::println);

}

}

Output; Before sorting:

dairymilk-Rs. 20

Thumsup-Rs. 100

Lays-Rs. 30

After sorting by price(descending):

dairymilk-Rs. 20

Lays-Rs. 30

Thumsup-Rs. 100

Sorted by price (ascending):

Thumsup-Rs. 100

Lays-Rs. 30

dairymilk-Rs. 20

Sorted by name (alphabetically):

dairymilk-Rs. 20

Lays-Rs. 30

Thumsup-Rs. 100

Q8. Sort Employee objects by joining date using Comparator.

Use Comparator to sort employees based on LocalDate or Date.

Code; **package** Polymorphism\_practice;

**import** java.util.\*;

**import** java.lang.\*;

**import** java.time.LocalDate;

**class** Employee4{

String name;

LocalDate date;

Employee4(String name,LocalDate date){

**this**.name=name;

**this**.date=date;

}

**public** LocalDate getdate() {

**return** date;

}

**public** String toString() {

**return** name+" joined on "+date;

}

}

**class** JoiningDateComparator **implements** Comparator<Employee4> {

**public** **int** compare(Employee4 e1, Employee4 e2) {

**return** e1.getdate().compareTo(e2.getdate());

}

}

**public** **class** Sort\_Employee\_objects {

**public** **static** **void** main(String[] args) {

List<Employee4> employees = Arrays.*asList*(

**new** Employee4("Ramesh",LocalDate.*of*(2023, 03, 10)),

**new** Employee4("Rajesh",LocalDate.*of*(2023, 04, 20)),

**new** Employee4("Ravi",LocalDate.*of*(2023, 02, 10)));

System.***out***.println("Before sorting:");

employees.forEach(System.***out***::println);

employees.sort(**new** JoiningDateComparator());

System.***out***.println("\nAfter sorting by joining date:");

employees.forEach(System.***out***::println);

}

}

Output; Before sorting:

Ramesh joined on 2023-03-10

Rajesh joined on 2023-04-20

Ravi joined on 2023-02-10

After sorting by joining date:

Ravi joined on 2023-02-10

Ramesh joined on 2023-03-10

Rajesh joined on 2023-04-20

Q9. Write a program that sorts a list of cities by population using Comparator.

Code; **package** Polymorphism\_practice;

**import** java.util.\*;

**class** cities{

String city\_name;

**int** population;

cities(String city\_name,**int** population){

**this**.city\_name=city\_name;

**this**.population=population;

}

**public** **int** getpopulation() {

**return** population;

}

**public** String toString() {

**return** city\_name +" with "+population;

}

}

**class** population\_cityComparator **implements** Comparator<cities>{

**public** **int** compare(cities c1, cities c2) {

**return** Integer.*compare*(c1.getpopulation(),c2.getpopulation());

}

}

**public** **class** sort\_cities\_population {

**public** **static** **void** main(String[] args) {

List<cities> city= Arrays.*asList*(

**new** cities("Mumbai",500000),

**new** cities("Hyderabad",6000),

**new** cities("Pune",4000));

System.***out***.println("Before sorting:");

city.forEach(System.***out***::println);

city.sort(**new** population\_cityComparator());

System.***out***.println("\nAfter sorting by population:");

city.forEach(System.***out***::println);

}

}

Output; Before sorting:

Mumbai with 500000

Hyderabad with 6000

Pune with 4000

After sorting by population:

Pune with 4000

Hyderabad with 6000

Mumbai with 500000

Q10. Use an anonymous inner class to sort a list of strings by length.

Code; **package** BAsicsof\_oops;

**import** java.util.\*;

**public** **class** SortByLength {

**public** **static** **void** main(String[] args) {

List<String> names = Arrays.*asList*("Apple", "Banana", "Kiwi", "Strawberry", "Fig");

Collections.*sort*(names, **new** Comparator<String>() {

@Override

**public** **int** compare(String s1, String s2) {

**return** Integer.*compare*(s1.length(), s2.length());

}

});

System.***out***.println("Sorted by length:");

**for** (String name : names) {

System.***out***.println(name);

}

}

}

Output; Sorted by length:

Fig

Kiwi

Apple

Banana

Strawberry

Q11. Create a program where:

Student implements Comparable to sort by name

Use Comparator to sort by marks

Demonstrate both sorting techniques in the same program.

Code; **package** BAsicsof\_oops;

**import** java.util.\*;

**import** java.lang.Comparable;

**class** Student\_n **implements** Comparable<Student\_n> {

**private** String name;

**private** **int** marks;

**public** Student\_n(String name,**int** marks){

**this**.name=name;

**this**.marks=marks;

}

**public** String getname() {

**return** name;

}

**public** **int** getmarks() {

**return** marks;

}

**public** String toString() {

**return** name+" obtained "+marks;

}

**public** **int** compareTo(Student\_n other) {

**return** **this**.name.compareTo(other.name);

}

}

**class** sort\_by\_marksComparator **implements** Comparator<Student\_n>{

**public** **int** compare(Student\_n s1,Student\_n s2) {

**return** Integer.*compare*(s1.getmarks(), s2.getmarks());

}

}

**public** **class** Sort\_by\_name\_marks {

**public** **static** **void** main(String[] args) {

List<Student\_n> students=Arrays.*asList*(

**new** Student\_n("vijay",79),

**new** Student\_n("Raju",80),

**new** Student\_n("suresh",95));

System.***out***.println("Before sorting:");

students.forEach(System.***out***::println);

Collections.*sort*( students);

System.***out***.println("\nAfter sorting by name:");

students.forEach(System.***out***::println);

System.***out***.println("\nAfter sorting by marks:");

students.sort(**new** sort\_by\_marksComparator());

students.forEach(System.***out***::println);

}

}

Output; Before sorting:

vijay obtained 79

Raju obtained 80

suresh obtained 95

After sorting by name:

Raju obtained 80

suresh obtained 95

vijay obtained 79

After sorting by marks:

vijay obtained 79

Raju obtained 80

suresh obtained 95

Q12. Sort a list of Book objects using both Comparable (by ID) and Comparator (by title, then author).

Code; **package** BAsicsof\_oops;

**import** java.util.\*;

**class** Book **implements** Comparable<Book>{

**int** id;

String title;

String author;

Book(**int** id,String title,String author){

**this**.id=id;

**this**.title=title;

**this**.author=author;

}

**public** **int** getid() {

**return** id;

}

**public** String gettitle() {

**return** title;

}

**public** String getauthor() {

**return** author;

}

**public** String toString() {

**return** id+" "+title+" "+author;

}

**public** **int** compareTo(Book other) {

**return** Integer.*compare*(**this**.id, other.id);

}

}

**class** TitleAuthorComparator **implements** Comparator<Book> {

@Override

**public** **int** compare(Book b1, Book b2) {

**int** titleCompare = b1.gettitle().compareTo(b2.gettitle());

**if** (titleCompare != 0) {

**return** titleCompare;

} **else** {

**return** b1.getauthor().compareTo(b2.getauthor());

}

}

}

**public** **class** Sort\_list\_books {

**public** **static** **void** main(String[] args) {

List<Book> book=Arrays.*asList*(

**new** Book(6,"wingsoffire","Dr.Abdul kalam"),

**new** Book(18,"The rise of hit","Rohit sharma"),

**new** Book(4,"Mylifemyrule","jagadeesh"));

Collections.*sort*(book);

System.***out***.println("Sorted by ID:");

**for** (Book b : book) {

System.***out***.println(b);

}

Collections.*sort*(book, **new** TitleAuthorComparator());

System.***out***.println("\nSorted by Title then Author:");

**for** (Book b : book) {

System.***out***.println(b);

}

}

}

Output; Sorted by ID:

4 Mylifemyrule jagadeesh

6 wingsoffire Dr.Abdul kalam

18 The rise of hit Rohit sharma

Sorted by Title then Author:

4 Mylifemyrule jagadeesh

18 The rise of hit Rohit sharma

6 wingsoffire Dr.Abdul kalam

Q13. Write a menu-driven program to sort Employee objects by name, salary, or department using Comparator.

Code; **package** BAsicsof\_oops;

**import** java.util.\*;

**class** Employee2{

String name;

**int** salary;

String depart;

Employee2(String name,**int** salary,String depart){

**this**.name=name;

**this**.salary=salary;

**this**.depart=depart;

}

**public** String getname() {

**return** name;

}

**public** **int** getsalary() {

**return** salary;

}

**public** String getdepart() {

**return** depart;

}

**public** String toString() {

**return** name+" "+salary+" "+depart;

}

}

**class** NameComparator **implements** Comparator<Employee2> {

**public** **int** compare(Employee2 e1, Employee2 e2) {

**return** e1.getname().compareTo(e2.getname());

}

}

**class** SalaryComparator **implements** Comparator<Employee2> {

**public** **int** compare(Employee2 e1, Employee2 e2) {

**return** Integer.*compare*(e1.getsalary(), e2.getsalary());

}

}

**class** DepartComparator **implements** Comparator<Employee2>{

**public** **int** compare(Employee2 e1,Employee2 e2) {

**return** e1.getdepart().compareTo(e2.getdepart());

}

}

**public** **class** Sort\_employee\_name {

**public** **static** **void** main(String[] args) {

List<Employee2> employees=Arrays.*asList*(

**new** Employee2("Raju",25000,"Developer"),

**new** Employee2("Ravi",15000,"Tester"),

**new** Employee2("somesh",35000,"Management"),

**new** Employee2("akash",45000,"HR"));

Scanner sc = **new** Scanner(System.***in***);

**int** choice;

**do** {

System.***out***.println("\nChoose sorting option:");

System.***out***.println("1. Sort by Name");

System.***out***.println("2. Sort by Salary");

System.***out***.println("3. Sort by Department");

System.***out***.println("4. Exit");

System.***out***.print("Enter your choice: ");

choice = sc.nextInt();

**switch** (choice) {

**case** 1:

Collections.*sort*(employees, **new** NameComparator());

System.***out***.println("\nSorted by Name:");

**break**;

**case** 2:

Collections.*sort*(employees, **new** SalaryComparator());

System.***out***.println("\nSorted by Salary:");

**break**;

**case** 3:

Collections.*sort*(employees, **new** DepartComparator());

System.***out***.println("\nSorted by Department:");

**break**;

**case** 4:

System.***out***.println("Exiting program.");

**break**;

**default**:

System.***out***.println("Invalid choice. Try again.");

}

**if** (choice >= 1 && choice <= 3) {

**for** (Employee2 e : employees) {

System.***out***.println(e);

}

}

} **while** (choice != 4);

sc.close();

}

}

Output;

Choose sorting option:

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter your choice: 2

Sorted by Salary:

Ravi 15000 Tester

Raju 25000 Developer

somesh 35000 Management

akash 45000 HR

Choose sorting option:

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter your choice: 1

Sorted by Name:

Raju 25000 Developer

Ravi 15000 Tester

akash 45000 HR

somesh 35000 Management

Choose sorting option:

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter your choice: 3

Sorted by Department:

Raju 25000 Developer

akash 45000 HR

somesh 35000 Management

Ravi 15000 Tester

Choose sorting option:

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter your choice: 1 3

Sorted by Name:

Raju 25000 Developer

Ravi 15000 Tester

akash 45000 HR

somesh 35000 Management

Choose sorting option:

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter your choice:

Sorted by Department:

Raju 25000 Developer

akash 45000 HR

somesh 35000 Management

Ravi 15000 Tester

Choose sorting option:

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter your choice: 4

Exiting program.

Q14. Use Comparator.comparing() with method references to sort objects in Java 8+.

Code; **package** BAsicsof\_oops;

**import** java.util.\*;

**class** Employee4 {

**private** String name;

**private** **int** salary;

**private** String depart;

**public** Employee4(String name, **int** salary, String depart) {

**this**.name = name;

**this**.salary = salary;

**this**.depart = depart;

}

**public** String getName() {

**return** name;

}

**public** **int** getSalary() {

**return** salary;

}

**public** String getDepart() {

**return** depart;

}

**public** String toString() {

**return** name + " " + salary + " " + depart;

}

}

**public** **class** SortWithComparatorComparing {

**public** **static** **void** main(String[] args) {

List<Employee4> employees = Arrays.*asList*(

**new** Employee4("Raju", 25000, "Developer"),

**new** Employee4("Ravi", 15000, "Tester"),

**new** Employee4("Somesh", 35000, "Management"),

**new** Employee4("Akash", 45000, "HR")

);

employees.sort(Comparator.*comparing*(Employee4::getName));

System.***out***.println("Sorted by Name:");

employees.forEach(System.***out***::println);

employees.sort(Comparator.*comparingInt*(Employee4::getSalary));

System.***out***.println("\nSorted by Salary:");

employees.forEach(System.***out***::println);

employees.sort(Comparator.*comparing*(Employee4::getDepart));

System.***out***.println("\nSorted by Department:");

employees.forEach(System.***out***::println);

}

}

Output; Sorted by Name:

Akash 45000 HR

Raju 25000 Developer

Ravi 15000 Tester

Somesh 35000 Management

Sorted by Salary:

Ravi 15000 Tester

Raju 25000 Developer

Somesh 35000 Management

Akash 45000 HR

Sorted by Department:

Raju 25000 Developer

Akash 45000 HR

Somesh 35000 Management

Ravi 15000 Tester

Q15. Use TreeSet with a custom comparator to sort a list of persons by age.

Code; **package** BAsicsof\_oops;

**import** java.util.\*;

**class** Person{

**int** age;

String name;

Person(**int** age,String name){

**this**.age=age;

**this**.name=name;

}

**public** **int** getage() {

**return** age;

}

**public** String getname() {

**return** name;

}

**public** String toString() {

**return** age+" "+name;

}

}

**class** AgeComparator **implements** Comparator<Person>{

**public** **int** compare(Person p1,Person p2) {

**return** Integer.*compare*(p1.getage(),p2.getage());

}

}

**public** **class** Treeset\_sort\_list\_persons {

**public** **static** **void** main(String[] args) {

TreeSet<Person> persons=**new** TreeSet<>(**new** AgeComparator());

persons.add(**new** Person(27,"Suresh"));

persons.add(**new** Person(45,"Naresh"));

persons.add(**new** Person(35,"Mahesh"));

System.***out***.println("Persons sorted by age:");

**for** (Person p : persons) {

System.***out***.println(p);

}

}

}

Output; Persons sorted by age:

27 Suresh

35 Mahesh

45 Naresh

**Q1. Create and Write to a File**

Write a Java program to create a file named student.txt and write 5 lines of student names using FileWriter.

Code; **package** File\_Handling\_practice;

**import** java.io.File;

**import** java.io.IOException;

**import** java.io.FileWriter;

**import** java.io.BufferedWriter;

**public** **class** Create\_Write\_file {

**public** **static** **void** main(String[] args) {

File f= **new** File("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\student.txt");

**try** {

**if**(f.createNewFile()) {

System.***out***.println("File created:"+f.getName());

}**else** {

System.***out***.println("File already exists.");

}

BufferedWriter b=**new** BufferedWriter(**new** FileWriter("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\student.txt"));

b.append("Raju");

b.newLine();

b.append("Ramesh");

b.newLine();

b.append("Somesh");

b.newLine();

b.append("Prabhas");

b.newLine();

b.append("Mahesh");

b.newLine();

b.append("Suresh");

b.close();

System.***out***.println("Names are written into file successfully.");

}

**catch**(IOException e) {

System.***out***.println(e);

}

}

}

Output; File already exists.

Names are written into file successfully.

**Q2. Read from a File**

Write a program to read the contents of student.txt and display them line by line using BufferedReader.

Code; **package** File\_Handling\_practice;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** read\_from\_file {

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

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\student.txt"));

List<String> lines=**new** ArrayList<String>();

String line;

**while**((line=br.readLine())!=**null**) {

lines.add(line);

}

**for**(String l:lines) {

System.***out***.println(l);

}

br.close();

}

}

Output; Raju

Ramesh

Somesh

Prabhas

Mahesh

Suresh

**Q3. Append Data to a File**

Write a Java program to append a new student name to the existing student.txt file without overwriting existing data.

Code; **package** File\_Handling\_practice;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.util.Scanner;

**public** **class** Append\_data\_to\_a\_file {

**public** **static** **void** main(String[] args) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.print("Enter the student name to append: ");

String studentName = scan.nextLine();

String filePath = "C:\\\\Users\\\\tatip\\\\OneDrive\\\\Desktop\\\\New folder\\\\student.txt";

**try** (FileWriter writer = **new** FileWriter(filePath, **true**)) {

writer.write(studentName + System.*lineSeparator*());

System.***out***.println("Student name appended successfully.");

} **catch** (IOException e) {

System.***out***.println("An error occurred while appending to the file.");

e.printStackTrace();

}

scan.close();

}

}

Output; Enter the student name to append: allu arjun

Student name appended successfully.

**Q4. Count Words and Lines**

Write a program to count the number of words and lines in a given text file notes.txt.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**public** **class** Count\_words\_lines {

**public** **static** **void** main(String[] args) {

**int** lineCount = 0;

**int** wordCount = 0;

**try** {

BufferedReader reader = **new** BufferedReader(**new** FileReader("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\notes.txt"));

String line;

**while** ((line = reader.readLine()) != **null**) {

lineCount++;

String[] words = line.trim().split("\\s+");

wordCount += words.length;

}

reader.close();

System.***out***.println("Total Lines: " + lineCount);

System.***out***.println("Total Words: " + wordCount);

}

**catch**(IOException e) {

System.***out***.println("An error occurred while reading the file.");

e.printStackTrace();

}

}

}

Output; Total Lines: 2

Total Words: 5

**Q5. Copy Contents from One File to Another**

Write a program to read from source.txt and write the same content into destination.txt.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**public** **class** Copy\_onefile\_another {

**public** **static** **void** main(String[] args) {

**try** {

BufferedReader reader=**new** BufferedReader(**new** FileReader("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\source.txt"));

BufferedWriter writer=**new** BufferedWriter(**new** FileWriter("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\destination.txt"));

String line;

**while** ((line = reader.readLine()) != **null**) {

writer.write(line);

writer.newLine();

}

System.***out***.println("File copied successfully.");

reader.close();

writer.close();

}**catch**(IOException e) {

System.***out***.println("Error occured while copying file");

e.printStackTrace();

}

}

}

Output; File copied successfully.

**Q6. Check if a File Exists and Display Properties**

Create a program to check if report.txt exists. If it does, display its:

* Absolute path
* File name
* Writable (true/false)
* Readable (true/false)
* File size in bytes

Code; **package** File\_Handling\_practice;

**import** java.io.File;

**public** **class** fileDetails {

**public** **static** **void** main(String[] args) {

//access existing file and print file name,location,length

File f=**new** File("C:\\\\Users\\\\tatip\\\\OneDrive\\\\Desktop\\\\New folder\\\\report.txt");

**if**(f.exists()) {

System.***out***.println("File name:"+f.getName());//print file name

System.***out***.println("File Location:"+f.getAbsolutePath());//file path

System.***out***.println("File size:"+f.length());//print length of size of file

System.***out***.println("File readable:"+f.canRead());

System.***out***.println("File Writable:"+f.canWrite());

}**else** {

System.***out***.println("File not found");

}

}

}

Output; File name:report.txt

File Location:C:\Users\tatip\OneDrive\Desktop\New folder\report.txt

File size:40

File readable:true

File Writable:true

**Q7. Create a File and Accept User Input**

Accept input from the user (using Scanner) and write the input to a file named userinput.txt.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** Createafile\_userinput {

**public** **static** **void** main(String[] args) {

**try** {

File f=**new** File("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\userinput.txt");

f.createNewFile();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter a string :");

String s=scan.nextLine();

BufferedWriter w=**new** BufferedWriter(**new** FileWriter(f));

w.write(s);

System.***out***.println("User input is succesfully written into file. ");

}**catch**(IOException e) {

System.***out***.println("File is not exists.");

e.printStackTrace();

}

}

}

FileOutput; Enter a string :

jagadeesh

User input is succesfully written into file.

**Q8. Reverse File Content**

Write a program to read a file data.txt and create another file reversed.txt containing the lines in reverse order.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Reversefile\_data {

**public** **static** **void** main(String[] args) {

String inputPath = "C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\data.txt";

String outputPath = "C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\reversed.txt";

**try** {

BufferedReader reader=**new** BufferedReader(**new** FileReader(inputPath));

List<String> lines = **new** ArrayList<>();

String line;

**while** ((line = reader.readLine()) != **null**) {

lines.add(line);

}

reader.close();

Collections.*reverse*(lines);

BufferedWriter writer=**new** BufferedWriter(**new** FileWriter(outputPath));

**for** (String reversedLine : lines) {

writer.write(reversedLine);

writer.newLine();

}

writer.close();

System.***out***.println("File reversed successfully!");

} **catch**(IOException e) {

System.***out***.println("An error occurred while processing the files.");

e.printStackTrace();

}

}

}

Output; File reversed successfully!

**Q9. Store Objects in a File using Serialization**

Create a Student class with id, name, and marks. Serialize one object and save it in a file named student.ser.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**class** Student **implements** Serializable{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**int** id;

String name;

**float** marks;

Student(**int** id,String name,**float** marks){

**this**.id=id;

**this**.name=name;

**this**.marks=marks;

}

**public** **int** getid() {

**return** id;

}

**public** String getname() {

**return** name;

}

**public** **float** getmarks() {

**return** marks;

}

**public** String toString() {

**return** id+" "+name+" "+marks;

}

}

**public** **class** Store\_objects\_serialization {

**public** **static** **void** main(String[] args) {

Student s1 = **new** Student(101, "Jagadheeswar", 92.5f);

**try** {

FileOutputStream fos = **new** FileOutputStream("student.ser");

ObjectOutputStream oos = **new** ObjectOutputStream(fos);

oos.writeObject(s1);

oos.close();

fos.close();

System.***out***.println("Student object serialized successfully!");

} **catch** (IOException e) {

System.***out***.println("Serialization failed.");

e.printStackTrace();

}

}

}

Output; Student object serialized successfully!

**Q10. Read Serialized Object from File**

Deserialize the student.ser file and display the object's content on the console.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**class** Student **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**int** id;

String name;

**float** marks;

Student(**int** id, String name, **float** marks) {

**this**.id = id;

**this**.name = name;

**this**.marks = marks;

}

**public** String toString() {

**return** "ID: " + id + ", Name: " + name + ", Marks: " + marks;

}

}

**public** **class** Read\_object\_deserialization {

**public** **static** **void** main(String[] args) {

**try** {

FileInputStream fis = **new** FileInputStream("student.ser");

ObjectInputStream ois = **new** ObjectInputStream(fis);

Student s = (Student) ois.readObject();

System.***out***.println("Deserialized Student:");

System.***out***.println(s);

ois.close();

fis.close();

} **catch** (IOException | ClassNotFoundException e) {

System.***out***.println("Deserialization failed.");

e.printStackTrace();

}

}

}

Output; Deserialized Student:

101 Jagadheeswar 92.5

**Q11. Print All Files in a Directory**

Write a program to list all files (not directories) inside a folder path given by the user.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**import** java.nio.file.Files;

**import** java.util.Scanner;

**public** **class** Print\_all\_files {

**public** **static** **void** main(String[] args) {

Scanner scan= **new** Scanner(System.***in***);

System.***out***.println("Enter folder path:");

String path=scan.nextLine();

File folder = **new** File(path);

**if** (folder.isDirectory()) {

File[] files = folder.listFiles();

System.***out***.println("Files inside the folder:");

**for** (File file : files) {

**if** (file.isFile()) {

System.***out***.println(file.getName());

}

}

} **else** {

System.***out***.println("The path entered is not a valid directory.");

}

scan.close();

}

}

Output; Enter folder path:

C:\Users\tatip\OneDrive\Desktop\New folder

Files inside the folder:

data.txt

destination.txt

notes of c.doc

notes of c.docx

notes.txt

report.txt

reserved.txt

reversed.txt

sample.txt

sample1.text

sample1.texts

sample1.txt

sample13.txt

source.txt

student.txt

test.txt

userinput.txt

win.docx

**Q12. Delete a File**

Write a program to delete a file (given by file name) if it exists.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** Delete\_file {

**public** **static** **void** main(String[] args) {

Scanner scan = **new** Scanner(System.***in***);

System.***out***.print("Enter the file name (with path if needed): ");

String fileName = scan.nextLine();

File f=**new** File(fileName);

**if**(f.exists()) {

f.delete();

System.***out***.println("File is deleted successfully.");

}**else** {

System.***out***.println("File does not exists.");

}

scan.close();

}

}

Output; File is deleted successfully.

**Q13. Word Search in a File**

Ask the user to enter a word and check whether it exists in the file notes.txt.

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** Word\_search\_file {

**public** **static** **void** main(String[] args) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter a word:");

String s=scan.nextLine();

File file=**new** File("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\notes.txt");

**boolean** found = **false**;

**if** (file.exists()) {

**try** (Scanner fileScanner = **new** Scanner(file)) {

**while** (fileScanner.hasNextLine()) {

String line = fileScanner.nextLine();

**if** (line.contains(s)) {

found = **true**;

**break**;

}

}

} **catch** (FileNotFoundException e) {

System.***out***.println("Error reading the file.");

}

**if** (found) {

System.***out***.println("The word \"" + s + "\" exists in the file.");

} **else** {

System.***out***.println("The word \"" + s + "\" was not found in the file.");

}

}

**else** {

System.***out***.println("File 'notes.txt' does not exist.");

}

scan.close();

}

}

Output; Enter a word:

Hii

The word "Hii" exists in the file.

**Q14. Replace a Word in a File**

Read content from story.txt, replace all occurrences of the word "Java" with "Python", and write the updated content to updated\_story.txt

Code; **package** File\_Handling\_practice;

**import** java.io.\*;

**public** **class** Replace\_word\_file {

**public** **static** **void** main(String[] args) {

**try** {

BufferedReader reader=**new** BufferedReader(**new** FileReader("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\story.txt"));

BufferedWriter writer = **new** BufferedWriter(**new** FileWriter("C:\\Users\\tatip\\OneDrive\\Desktop\\New folder\\updated\_story.txt"));

String line;

**while** ((line = reader.readLine()) != **null**) {

String updatedLine = line.replace("Java", "Python");

writer.write(updatedLine);

writer.newLine();

}

reader.close();

writer.close();

System.***out***.println("Replacement complete. Check 'updated\_story.txt'.");

}**catch**(IOException e) {

System.***out***.println("An error occurred during file processing.");

e.printStackTrace();

}

}

}

Output; Replacement complete. Check 'updated\_story.txt'.