VÍ DỤ:

2.1.4. Chức năng sắp xếp theo giá

**Kỹ thuật sử dụng:**

* Comparator<Car> là một interface trong Java, được sử dụng để định nghĩa cách so sánh hai đối tượng Car. Lớp này có 1 phương thức duy nhất là compare nhận 2 đối tượng và trả về 1 số nguyên.
* Sử dụng biểu thức Lamda để cung cấp 1 các viết ngắn gọn.
* Sử dụng phương thức sort của lớp Collections để sắp xếp danh sách results theo thứ tự được xác định bởi Comparator<Car> là carPriceComparator.
* Ta sử dụng 1 danh sách khác sao chép lại danh sách list vì không muốn thay đổi trên danh sách gốc. Khi ta thực hiện chức năng sắp xếp xong, ta có thể ấn nút “Refresh” để hiển thị lại toàn bộ danh sách list gốc lên Jtable.

**Code:**

|  |
| --- |
| // Tạo một Comparator để so sánh  Comparator<Car> carPriceComparator = (c1,c2) -> Double.*compare*(c1.getProduct\_price(), c2.getProduct\_price());  btn\_sort.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e) {  //Sắp xếp list  List<Car> results = **new** ArrayList<>(*list*);  Collections.*sort*(results,carPriceComparator);  System.***out***.println(results);  //Cập nhật lại DefaultTableModel  DefaultTableModel model1 = (DefaultTableModel) table.getModel();  model1.setRowCount(0); //Xoá tất cả các dòng trong JTable  **for**(Car car : results) {  model1.addRow(**new** Object[]{  car.getProduct\_id(),  car.getProduct\_name(),  car.getProduct\_price(),  car.getProduct\_total(),  car.getModel(),  car.getColor(),  car.getEngineCapacity()  });  }  }  }); |

LÀM TƯƠNG TỰ NHƯ VÍ DỤ TRÊN:

2.2. Quản lý tour du lịch

***Ý tưởng:***

* Tạo một ArrayList<Tour> list nhằm thực hiện các thao tác thêm, xóa, sửa,… với dữ liệu
* Dữ liệu được nhập từ file Tour.bin thông qua FileInputStream và ObjectInputStream
* Các chức năng chính bao gồm: Thêm, Xóa, Sửa, Tìm kiếm, Sắp xếp theo giá, Đề xuất.
* Sử dụng Swing để tạo giao diện với 2 phần chính: inputPanel là 1 JPanel để nhập dữ liệu và tableScrollPane là 1 JScrollPane để hiển thị dữ liệu có sẵn.

2.2.1. Chức năng thêm mới

**Kỹ thuật sử dụng:**

* Lấy dữ liệu từ TextField để khởi tạo 1 đối tượng Tour mới và thêm vào danh sách Tour
* Try catch để xử lí ngoại lệ, báo lỗi nếu có vấn đề trong quá trình thêm mới

**Code:**

|  |
| --- |
| addButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent e) {  **try**  {  **int** id = Integer.*parseInt*(idField.getText());  String name = nameField.getText();  **int** price = (**int**)Double.*parseDouble*(priceField.getText());  **int** total = Integer.*parseInt*(totalField.getText());  String destination = destinationField.getText();  String depature = departureField.getText();  **int** duration = Integer.*parseInt*(durationField.getText());    Tour tour = **new** Tour(destination, depature, duration, id, name, price, total);  **if**(tourManager.addTour(tour))  {  tableModel.addRow(**new** Object[]{tour.getProduct\_id(), tour.getProduct\_name(), tour.getProduct\_price(), tour.getProduct\_total(),  tour.getDestination(), tour.getDeparture\_date(), tour.getDuration()});  }  }**catch**(Exception ex) {  System.***out***.println("Error");  System.***out***.println(ex);  }  }  }); |

|  |
| --- |
| **public** **boolean** addTour(Tour t) {  **boolean** find = **false**;  **for**(Product item: **this**.list) {  **if**(item.getProduct\_id() == t.getProduct\_id()) {  find = **true**;  **break**;  }  }  **if**(!find) {  **this**.list.add(t);  setCount(getCount()+1);  **return** **true**;  }**else** {  System.***out***.println("add failed!");  **return** **false**;  }  } |

2.2.2. Chức năng sửa thông tin

**Kỹ thuật sử dụng:**

* Lấy dữ liệu từ TextField để khởi tạo 1 đối tượng Tour mới và sửa Tour trong danh sách với id trùng với id của đối tượng truyền vào.
* Đối tượng cần sửa sẽ bị xóa và thay thế bằng đối tượng mới có thông tin muốn sửa thành, sau đó hiển thị lên màn hình.

**Code:**

|  |
| --- |
| editButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent e) {  **try**  {  **int** id = Integer.*parseInt*(idField.getText());  String name = nameField.getText();  **int** price = (**int**)Double.*parseDouble*(priceField.getText());  **int** total = Integer.*parseInt*(totalField.getText());  String destination = destinationField.getText();  String depature = departureField.getText();  **int** duration = Integer.*parseInt*(durationField.getText());  Tour tour = **new** Tour(destination, depature, duration, id, name, price, total);  **if**(tourManager.editTour(tour))  {  displayAfter();  }  }**catch**(Exception ex) {  System.***out***.println("Error");  }    }  }); |

|  |
| --- |
| **public** **boolean** editTour(Tour t) {  **boolean** find = **false**;  **int** cnt = -1;  **for**(Product item: **this**.list) {  cnt++;  **if**(item.getProduct\_id() == t.getProduct\_id()) {  find = **true**;  **break**;  }  }  **if**(find) {  **this**.list.remove(cnt);  **this**.list.add(t);  **return** **true**;  }**else** {  System.***out***.println("edit failed!");  **return** **false**;  }  } |

|  |
| --- |
| **public** **void** addToTable() {  tourManager.list.forEach(tour ->{  tableModel.addRow(**new** Object[]{  tour.getProduct\_id(), tour.getProduct\_name(),  tour.getProduct\_price(), tour.getProduct\_total(),  tour.getDestination(), tour.getDeparture\_date(), tour.getDuration()});  });  }      **public** **void** displayAfter() {  tourManager.printInfo();    tableModel.setRowCount(0);    addToTable();  } |

2.2.3. Chức năng xoá

**Kỹ thuật sử dụng:**

* Lấy dữ liệu từ TextField tìm đối tượng tương ứng với id và truyền vào phương thức xóa.
* Phương thức xóa sẽ xóa 1 đối tượng trong list tương ứng với đối tượng được truyền vào.

**Code:**

|  |
| --- |
| removeButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent e) {  **try**  {  **int** selectedRow = tourTable.getSelectedRow();  **if** (selectedRow >= 0) {  String id = tourTable.getModel().getValueAt(selectedRow, 0).toString();  Tour tour = **null**;    **for**(Product d : tourManager.list) {  **if**(d.getProduct\_id() == Integer.*parseInt*(id)) tour = (Tour) d;  }    **if**(tourManager.delTour(tour)) {  tableModel.removeRow(selectedRow);  }    }  }**catch**(Exception ex) {  System.***out***.println("Error");  }    }  }); |

|  |
| --- |
| **public** **boolean** delTour(Tour t) {    setCount(getCount()-1);  **if**(**this**.list.remove(t)) {  **this**.printInfo();  System.***out***.println("");  **return** **true**;  }  **return** **false**;  } |

2.2.4. Chức năng sắp xếp theo giá

**Kỹ thuật sử dụng:**

* Sắp xếp lại danh sách Tour theo giá bằng Collections.sort và phương thức tourPriceComparator. Sắp xếp tăng dần hay giảm dần sẽ phụ thuộc vào việc nhấn nút Sort desc hay Sort asc.

**Code:**

|  |
| --- |
| Comparator<Tour> tourPriceComparator = (c1,c2) -> Double.*compare*(c1.getProduct\_price(), c2.getProduct\_price());  @Override  **public** List<Tour> sortedTour(**double** price) {  **if**(price!=0) {  Collections.*sort*(**this**.list, tourPriceComparator);  }**else** {  Collections.*sort*(**this**.list, tourPriceComparator.reversed());  }  **try** {  **this**.exportFile("Tour.bin");  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  **return** **this**.list;  } |

2.2.5. Chức năng tìm kiếm theo tên/ giá

**Kỹ thuật sử dụng:**

* Lấy dữ liệu bao gồm keyword và search option từ TextField.
* Keyword dùng để tìm kiếm thì sẽ được dùng để khởi tạo 1 đối tượng ByteArrayInputStream nhằm thiết lập luồng đầu vào chuẩn của hệ thống (System.in) thành đối tượng vừa tạo.
* Tạo stream để xử lí các phần tử trong danh sách Tour.
* Lọc các phần tử theo điều kiện nhất định.
* Ép đối tượng sang kiểu Tour.
* Tạo list dựa trên các phần tử đã lọc ra.

**Code:**

|  |
| --- |
| searchButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent e) {  **try** {  String data = "" + cb.getItemAt(cb.getSelectedIndex());    **int** searchOpt = cb.getSelectedIndex()+1;  System.***out***.println("Search by "+data);  String keyword = searchField.getText();    ByteArrayInputStream inp = **new** ByteArrayInputStream(keyword.getBytes());  System.*setIn*(inp);    List<Tour> res = tourManager.searchTour(searchOpt+"");  displaySearchResult(res);  res.forEach(go ->{  System.***out***.println(go.toString());  });  }**catch**(Exception ex) {  System.***out***.println("Error");  }  }  }); |

|  |
| --- |
| **public** List<Tour> searchTour(String name) {  List<Tour> res = **new** ArrayList<>();  Scanner scanner = **new** Scanner(System.***in***);  System.***out***.println("keyword:");  String keyword = scanner.nextLine();  **switch**(name) {  **case** "1":  {  System.***out***.println("find by name result:");  res = list.stream()  .filter(p -> p **instanceof** Tour)  .map(p -> (Tour) p)  .filter(t -> t.getProduct\_name().toLowerCase().contains(keyword.toLowerCase()))  .collect(Collectors.*toList*());  **break**;  }  **case** "2":  {  System.***out***.println("find by price result:");  **final** **double**[] keywordDouble = { Double.*parseDouble*(keyword) };  res = list.stream()  .filter(p -> p **instanceof** Tour)  .map(p -> (Tour) p)  .filter(t -> t.getProduct\_price() == keywordDouble[0])  .collect(Collectors.*toList*());  **break**;  }  **case** "3":  {  System.***out***.println("find by destination result:");  res = list.stream()  .filter(p -> p **instanceof** Tour)  .map(p -> (Tour) p)  .filter(t -> t.getDestination().toLowerCase().contains(keyword.toLowerCase()))  .collect(Collectors.*toList*());  **break**;  }  **case** "4":  {  System.***out***.println("find by departure date result:");  res = list.stream()  .filter(p -> p **instanceof** Tour)  .map(p -> (Tour) p)  .filter(t -> t.getDeparture\_date().equalsIgnoreCase(keyword))  .collect(Collectors.*toList*());  **break**;  }  **case** "5":  {  System.***out***.println("find by duration result:");  res = list.stream()  .filter(p -> p **instanceof** Tour)  .map(p -> (Tour) p)  .filter(t -> (t.getDuration()+"").equalsIgnoreCase(keyword))  .collect(Collectors.*toList*());  **break**;  }  }    **return** res;  } |

|  |
| --- |
| **public** **void** displaySearchResult(List<Tour> res){    tableModel.setRowCount(0);  res.forEach(tour ->{  tableModel.addRow(**new** Object[]{tour.getProduct\_id(), tour.getProduct\_name(),  tour.getProduct\_price(), tour.getProduct\_total(),  tour.getDestination(), tour.getDeparture\_date(), tour.getDuration()});  });  } |

2.2.6. Các chức năng khác

**Kỹ thuật sử dụng:**

* Chức năng đề xuất Tour được thực hiện khi người dùng chọn 1 Tour bất kì và nhấn nút Rec.
* Lấy dữ liệu từ TextField bao gồm price, destination, departureDate và duration để tìm Tour tương tự Tour được chọn
* Lặp qua danh sách Tour 1 lần để tìm ra các Tour có tiêu chí phù hợp với Tour được chọn. Các tiêu chí này là do người lập trình đề ra.

**Code:**

|  |
| --- |
| recButton.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e)  {  **try**  {  **int** selectedRow = tourTable.getSelectedRow();  **if** (selectedRow >= 0)  {  **double** price = Double.*parseDouble*(tourTable.getModel().getValueAt(selectedRow, 2).toString());  String destination = tourTable.getModel().getValueAt(selectedRow, 4).toString();  String depature = tourTable.getModel().getValueAt(selectedRow, 5).toString();  **int** duration = Integer.*parseInt*(tourTable.getModel().getValueAt(selectedRow, 6).toString());  String[] dateselected = depature.split("/");  List<Tour> recTour = **new** ArrayList<Tour>();  **for**(Tour go : tourManager.list) {  **if**(!go.getDestination().equalsIgnoreCase(destination))  {  **continue**;  }  **else**  {  **boolean** dk1 = (go.getProduct\_price()> price\*0.7 && go.getProduct\_price() <= price);  **boolean** dk2 = (go.getDuration() < duration + 3) && (go.getDuration() > duration\*0.5);  String[] dateinloop = go.getDeparture\_date().split("/");  **boolean** dk3 = (dateinloop[2].equals(dateselected[2])) && (dateinloop[1].equals(dateselected[1])) &&  (  Integer.*parseInt*(dateselected[0])+3 > Integer.*parseInt*(dateinloop[0])&& Integer.*parseInt*(dateselected[0])-3 < Integer.*parseInt*(dateinloop[0])  );    **if** ((dk1 ? 1 : 0)+(dk2 ? 1 : 0)+(dk3 ? 1 : 0) >= 2)  {  recTour.add(go);  }  }  }  displaySearchResult(recTour);  }  }  **catch**(Exception ex)  {  System.***out***.println("Error");  }  }  }); |