PRACTICAL CASE

Digital Navigation

Human-Computer Interfaces

IPC – DSIC

UPV

Curso 2024-2025

Index

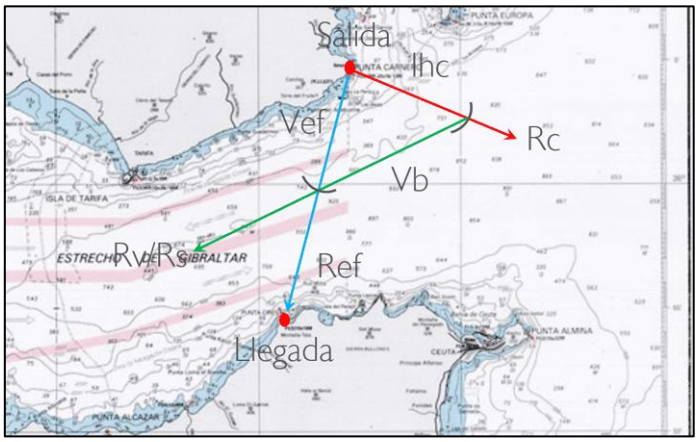
1. Case Study

The aim is to develop a tool to facilitate the preparation of navigation exercises using the nautical chart of the Strait of Gibraltar for recreational boat skipper examinations. This process is described below.

The exams for this degree are multiple choice. Students are presented with a problem statement and are given four answers, one of which is correct.

> 32. - From the position l = 35º45.0´N L = 006º09.8´W, we navigate on the effective course Ref = 042º for one hour, reaching the new position l = 35º52.6´N - L = 006º01.5´W. What is the distance travelled between the two positions?
>
> A) d = 12,6'
>
> B) d = 10,2'
>
> C) d = 9,2'
>
> D) d = 11,2'

The student uses the nautical chart to make the necessary calculations and thus be able to choose the correct answer.



On the nautical chart the student can perform the following operations:

- plot points,
- draw lines between two points,
- draw circles whose center is one of these points.
- Take the distances between two points, then transfer this measurement onto the chart's vertical scale to obtain the actual distance in miles.
- Write on the chart intermediate calculations or notes useful for solving the problem.

For all this, you have a chart, a pencil, an eraser, a ruler to draw lines and measure distances between two points, a compass to draw arcs, and a square protractor.

Therefore, the application to be developed aims to digitize this process in order to avoid the use of the paper nautical chart. This way, the student will be able to practice using their computer, although the final exam will always be conducted with the paper chart and the described tools.

Thus, there are two groups of functions in the application to be developed. First, there is a set of features that provide the student with problem statements along with a set of possible answers. These features will also record the student's correct and incorrect answers and keep a history of their actions.

To achieve this, the user will register and create a user profile. Each time they start a session, the application will propose problems that the user must solve. At the end of the session, the correct and incorrect answers will be saved. This stored information can later be viewed to track the student's learning progress.

On the other hand, the application will include the necessary functionality to draw on a digital chart: points, lines, arcs, and writing annotations. The tool aims to provide a digital environment similar to the physical one in which the student will take the exam — that is, having the same physical tools in a digital format. The student will also be able to calculate angles and measure distances.

Below are the use case scenarios obtained after analyzing the system requirements, which must be used to properly design and implement the required application. These scenarios have been grouped into two categories, as described above.

2. Scenarios about the user

2.1. Register on the app

Juan wants to prepare for the recreational boat skipper exam. At the academy where he enrolled, the instructor uses a digital tool that is available for download. After installing it, and in order to access the set of available solved problems, he selects the registration option.

The system asks him to enter a username (which will be used to identify him), a valid email address, a password, his date of birth, and optionally, an image to be used as an avatar. Juan enters "jgarcia" as his username. For the password, he chooses "passPER21!" and provides the email address: "jgarcia@gmail.com". Finally, he enters a date of birth that makes him appear slightly younger than he actually is. Since he doesn't feel like looking for an avatar at that moment, he keeps the default one provided. After entering all the required information, he proceeds to finalize the process.

The system verifies the data entered and notifies him that the username is already in use. It asks him to choose a new one that contains between 6 and 15 characters or digits, without spaces, and optionally allowing hyphens or underscores. Juan decides to go with the nickname "jpgarcia".

The system verifies that the rest of the data entered is valid. The password is between 8 and 20 characters long, including at least one uppercase letter, one lowercase letter, a digit, and a special character (!@#$%&*()-+=). The email address has a valid format and the user is over 16 years old. The system registers the user and informs him accordingly.

2.2. Authenticate

Juan has some free time and decides to do a few navigation exercises before going to the academy. He opens the application and selects the option to log in to the system. The system asks him to enter his username or nickname and his password. Juan enters his nickname and password. The system verifies that a user with those credentials exists and authenticates him, granting access to the rest of the application's features.

2.3. Log out

Juan is tired of doing exercises, and Susana wants to try to see if she can complete four exercises in a row correctly. Instead of closing the application, Juan logs out so that Susana can start her own session.

2.4. Perform a problem

After logging in, Juan asks the system to suggest a problem. Today, he doesn't feel like thinking too much, so he chooses to have the system propose a problem randomly. The system suggests a problem, displays the statement, and the four possible answers. Before showing the answers, the system always randomizes their order.

After performing the calculations on the chart, Juan selects the answer he believes is correct and asks the system to verify the solution.

On another day, Juan will ask the system to show him the list of problems and will choose one himself.

2.5. Modify profile

Juan didn't choose an avatar when he registered, but he's tired of having the default avatar, so he decides to access the option to modify his profile information. The

system displays the current information: his username, password, email address, date of birth, and the avatar assigned to him. Juan realizes he can modify any data except his username. He decides to change his avatar from a few images stored on his computer. After making the change, he requests that the information be updated.

The system verifies that all the changes meet the established requirements. The password contains between 8 and 20 characters, including at least one uppercase letter, one lowercase letter, a digit, and a special character (!@#$%&*()-+=). The email address has a valid format, and the user is over 12 years old. Therefore, the system updates the information and notifies the user.

2.6. Show results

Juan wants to know how he's progressing in the subject, so after logging in, he asks the application to show him information about the correct and incorrect answers he has given. The system displays his data in a structured format. Juan wants to see this information, but only for the last few days, so he filters the data to be displayed by specifying the day from which he wants the results to appear.

3. Scenarios on the Chart

Just as in the real-world environment where the user has pencils of different colors and tips of different thicknesses, in the actions described below that require it, there must be an option to choose the color and thickness of the stroke.

3.1. Zoom in

Juan is working on an exercise, but he can't clearly see the name of a lighthouse on the chart, so he zooms in to get a closer view. After reading the information, he decides to return to a more distant view and zooms out. The zoom is applied to the digital chart.

3.2. Plot a point

Juan decides to plot a point on the chart. He uses the point tool and selects the position on the chart. The system displays the point in the shape and color previously set.

3.3. Draw a line

Juan wants to draw a line that passes through two points, so he accesses the line drawing tool. After selecting the starting point and the ending point, the line is marked with the pre-established thickness and color.

3.4. Draw an arc

Juan wants to draw an arc, so he accesses the arc drawing tool. After selecting the center of the arc and its radius, the arc is marked with the pre-established thickness and color. This arc can have a free radius or a preset radius. In this case, Juan takes a measurement along the vertical axis of the chart.

3.5. Annotate text

Juan wants to add text on the chart. He selects the text tool and, after choosing the location where it will be displayed, he types the corresponding text. Once finished, the system displays the text with the pre-established color and size.

3.6. Change the color of a mark

Juan wants to change the color of a mark (point, line, arc, text), so he selects the color and the mark, and the system displays the mark with the new appearance.

3.7. Delete a mark

Juan is not happy with the last line he drew, so he selects it and deletes it from the chart. At another time, he also deletes a point, an arc, and some text.

3.8. Clean the chart

Juan has solved the problem and wants to clear the chart to work on another problem. He selects the clear option, and the system presents him with a chart without any marks.

3.9. Move the protractor to measure angles. (Take an angle / draw a line with an angle)

Juan wants to know the angle formed by a line, so he selects the protractor, and moves the protractor across the chart until its center is positioned over the line. Once he sees the intersection point of the line on the protractor, he knows the angle it forms with the north. Shortly after, he wants to draw a line from a lighthouse at a specific angle. To do this, he places the center of the protractor at the lighthouse. To draw the line, he just needs to mark a point on the chart near the desired angle measurement on the protractor.

Take an angle:

We can state that measuring an angle is a task composed of several of the scenarios described in point 3. That is, to annotate the angle of a line, the user will first have to draw the line, then move the protractor, and finally, annotate the text they deem appropriate. In the following link, you can access a video that shows an example of this high-level task: annotating an angle.

Draw a line with an angle:

Similar to the previous case, in the following video we show a high-level task that uses several scenarios: drawing a line with an angle

3.10. Take a distance on the chart

Juan wants to know the distance between two points, so he uses the available ruler. He moves the ruler until it connects the two points, and with the measurement taken, he moves the ruler to the vertical scale of the chart, where he manually calculates the

number of scale subdivisions that correspond to the centimeters measured with the ruler. Since the measurement in centimeters is not relevant, this process can also be done with a compass. By opening the compass with its tips at the two points to be measured, he can then move the compass to the vertical scale of the chart and calculate the number of scale subdivisions that this measurement spans.

3.11. Mark the ends of a point on the chart

Juan wants to know the latitude and longitude of a marked point on the chart. After selecting the appropriate tool and selecting the point, the system draws two orthogonal lines on the chart that pass through the point and intersect the edges of the chart perpendicularly. The latitude is the point of intersection on the vertical scale, and the longitude is the point of intersection on the horizontal scale.

4. Resources to do the practice

To complete the exercise, the file "carta_nautica.jpg" is provided, which will be used as the base to perform the different use cases.

The file "transportador.jpg" is also provided as a square angle protractor. For this protractor to be useful, it must be made transparent. JavaFX nodes have the property "**Opacity**," which allows you to adjust their transparency. It is not mandatory to use this resource; the student may use other strategies or sources to make the angle protractor transparent. The resource "regla.jpg" is also provided as a ruler, which will be used to take measurements or draw lines if necessary.

5. Data model

The study of the task objects has resulted in the objects: User, Session, Problem and Answer.

The Java implementation of these classes is detailed below:

User

User class allows you to manage information about system users. User attributes are:

- String **nickName**: Username. Cannot be updated.
- String **email**: user's email address.
- String **password:** user password.
- Image **avatar**: user avatar image.
- LocalDate **birthdate**: user's date of birth.
- ArrayList <Session> **sessions**: a list of all sessions that the user has started, which stores the number of problems solved correctly and the number of problems not solved correctly.

Session

The Session class stores the information of each session performed by the user. The attributes of a Session cannot be modified once it has been created. Therefore, the object should be created when the user logs out, either voluntarily or because the

application is closed. It is at this point that the number of correct or incorrect answers during the session is known.

The attributes of the Session class are:

- LocalDateTime **timeStamp**: day and time the session is started/recorded.
- int **hits**: number of problems solved correctly.
- int **faults**: number of problems solved incorrectly.

Problem

The Problem class is used to store navigation problems. The database includes a set of problems, so it is not necessary to create objects of this type. The fields of the class are:

- StringProperty **text**: property with the text of the problem statement.
- ArrayList <Answer> **answers**: list with the four answers, objects of type answer.

Answer

The Answer class is used to store an answer to a navigation problem. The database includes a set of problems with their answers, so it is not necessary to create objects of this type. The fields of the class are:

- StringProperty **text**: property with the response text.
- BooleanProperty **validity**: property with the certainty value of the answer (is it correct or not).

6. Data persistence

The persistence of information will be carried out through an SQLite[1] database. The project will provide the IPC2025.jar library. This library defines the data model described in the previous section, meaning you DO NOT need to code the classes described in the model; you must use the already implemented classes in this library. It will allow you to store and retrieve all the information that requires persistence. Specifically, the application will maintain a set of navigation problems with their answers. It will also store user information, their data, and for each session started. Moreover, this library will store the number of problems solved correctly and the number of problems that are not.

The library manages all access to the database, creating it automatically if necessary within your project directory in a file named "database.db". If the system creates the database, it will be empty, so there will be no problems to display to the user. We will provide a database that contains a set of problems and a generic user so you can test your work in case you haven't defined user registration yet.

[1] https://www.sqlite.org/index.html

A method is provided. It allows you to dump the contents of the database to a text file in case you need to verify its content during the project development. However, there are numerous applications that allow you to open an SQLite database and view the contents of its tables. For example, you can install the free application "*DB Browser for SQLite*" [2].

This library and the documentation for using it in the project will be published in a second PDF.

7. Base project

To complete the work, the NetBeans project, **Poi_UPV**, is provided. This project demonstrates how to implement the zoom functionality described in the task scenario and can be used as the base project for developing the work (it is not mandatory to start from this project).

To implement the zoom, an ImageView is used, which occupies an entire container. A map (similar to the nautical chart requested in the assignment) is displayed in the ImageView. The appropriate JavaFX container for applying zoom is the ScrollPane. This project shows how to include an image of a map in this container and then apply scaling. When the image size exceeds the container size due to scaling, scrollbars automatically appear, allowing us to move around the entire container. If objects have been added behind the ImageView, they will be displayed above it, and if we apply scaling to the container, it will also affect these objects. The scaling will not affect the position of the node within the map, as the scaling affects both nodes equally, meaning it will maintain its (x, y) position relative to the (0, 0) of the image.

For this strategy to work correctly, it is necessary to use objects of the Group type, which are not available in SceneBuilder. The necessary code is included in the initialize() method of the controller class and is explained through comments.

It is recommended to use this project as the base for the work, removing or changing any elements that are not suitable for the required functionality.

8. Delivery Instructions

The first steps to develop the submission involve creating the conceptual design and later the physical design using low-fidelity prototypes.

We recommend that you review the prototypes you create with your practical or theory professor. Keep in mind that in theory class, we have finished the conceptual design and are starting with the physical design, both of which are necessary to create the appropriate design for the submission.

In a later post, we will reference the methods from the IPC2025.jar library, as well as instructions for adding the library to the project.

We will also provide helper code to draw nodes on the chart and/or move them.

[2] https://sqlitebrowser.org/

The submission will be made in groups of 3 students, all belonging to the same laboratory group.