## Soal

*Case*

**SeaPort**

**Criteria**:

1. Abstract Class

You need to design at least **three** classes, **one abstract** class, and **two concrete** classes. Abstract class consists of all **common** attributes and behavior that both of concrete class had. Concrete class consist of **specific** attribute and behavior that not common between the concrete classes

1. Encapsulation

To **hide** the data of a class from an **illegal** direct access, all the attributes of the class must be **encapsulated** and will be accessed using an **accessor** and **mutator** that may perform validation before accessing the encapsulated attribute

1. Inheritance

All the concrete class **must inherit all** attribute and behavior from the abstract class

1. Polymorphism

If the concrete class has **a specific implementation** of the inherited behavior (method) that **differ** from the abstract class, the concrete class can **override** or **overload** the behavior from the abstract class

SeaPort is an international harbor where ships all over the world can port and get assigned to the port’s database. As more ships came in, the government asks you to make a program that can help the port’s manager to manage their data. The government wants you to make this program based on the following criteria:

1. **Menu**

When the program is run, the program will show this menu. In this menu, the user can choose whether they want to **insert a new ship**, **view all ships**, **update ship data**, or **exit the program**.

Text

Description automatically generated

Figure 1. Menu

1. **Insert new ship**

If the user chooses to **insert a new ship**, then ask the user what type of ship they want to insert. Validate that the ship type must **either be “cruiseship” or “battleship” (case insensitive)**.

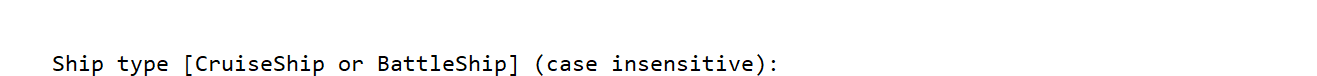


Figure 2. Input Ship Type to Insert

* If the user chooses to insert a new **cruiseship**, then ask for the cruiseship’s detail on the following criteria:
* Ask for the **cruiseship name**. Validate that the length of the cruiseship’s name must be **between 5 and 20 characters (inclusive)** and the name must **starts with ‘The ‘**.
* Ask for the **cruiseship total crew**. Validate that the cruiseship’s total crew must be **higher than 0**.
* Ask for the **cruiseship availability**. Validate that the input only can be **either ‘y’ or ‘n’ (case insensitive)**.
* Ask for the **cruiseship total passengers**. Validate that the cruiseship’s total passengers must be **higher than 0**.
* Ask for the **cruiseship company name**. Validate that the length of the cruiseship’s company name must be **between 5 and 20 characters (inclusive)**.
* Then, generate a **random** ID for the cruiseship based on the following format:

**ID = ‘CS[0-9][0-9][0-9]’**

**Example: CS123**

Validate that the generated ID must be **unique**.

* Finally, **add** the cruiseship to an **Array** / **ArrayList** / **Vector**.

Text

Description automatically generated

Figure 3. Insert Cruise ship

* If the user chooses to insert a new **battleship**, then ask for the battleship’s detail on the following criteria:
* Ask for the **battleship country name**. Validate that the length of the battleship’s country name must be **between 5 and 20 characters** **(inclusive)**.
* Ask for the **battleship name**. Validate that the length of the battleship’s name must be **between 5 and 20 characters (inclusive)** and the name must **end with ‘ [first letter of company name]MS’**.
* Ask for the **battleship total crew**. Validate that the battleship’s total crew must be **higher than 0**.
* Ask for the **battleship availability**. Validate that the input only can be **either ‘y’ or ‘n’ (case insensitive)**.
* Ask for the **battleship total troops**. Validate that the battleship’s total troops must be **higher than 0**.
* Then, generate a **random** ID for the battleship based on the following format:

**ID = ‘BS[0-9][0-9][0-9]’**

**Example: BS123**

Validate that the generated ID must be **unique**.

* Finally, **add** the battleship to an **Array** / **ArrayList** / **Vector**.

Text

Description automatically generated

Figure 4. Insert Battleship

1. **View all ships**

If the user chooses to **view all the ships**, then validate the view menu based on the following criteria:

* If there are **no** ships, then inform the user that there are no ships yet.



Figure 5. View All Ships with No Ships

* Otherwise, view all the ships at once in two tables. The first table will display all cruise ships and the second table will display all battleships.

Table

Description automatically generated

Figure 6. View All Ships

1. **Update ship data**

If the user chooses to **update ship data**,then validate the update menu based on the following criteria:

* If there are **no** ships, then inform the user that there are no ships yet.



Figure 7. View All Ships with No Ships

* Otherwise, ask the user what type of ship they want to update. Provide a feature for the user to update a **Cruiseship** or **Battleship**.

Text, letter

Description automatically generated

Figure 8. Choose Ship Type to Update

* If the user chooses to update a **cruiseship**, then the program will do the following:
* First, display all existing cruiseships and ask which cruiseship they want to update. Validate that the user can only input the number from **1 to the amount of cruiseships that exist**.
* Ask for the **cruiseship new name**. Validate that the length of the cruiseship’s name must be **between 5 and 20 characters (inclusive)** and the name must **start with ‘The ‘**.
* Ask for the **cruiseship new total crew**. Validate that the cruiseship’s total crew must be **higher than 0**.
* Ask for the **cruiseship new availability**. Validate that the input only can be **either ‘y’ or ‘n’ (case insensitive)**.
* Ask for the **cruiseship new total passengers**. Validate that the cruiseship’s total passengers must be **higher than 0**.
* Ask for the **cruiseship new company name**. Validate that the length of the cruiseship’s company name must be **between 5 and 20 characters (inclusive)**.
* Finally, **update** the selected cruiseship’s data with the new cruiseship’s data.

Text

Description automatically generated

Figure 9. Update Cruise ship

* If the user chooses to update a **battleship**, then the program will do the following:
* First, display all existing battleships and ask which battleship do they want to update. Validate that the user can only input the number from **1 to the amount of battleships that exist**.
* Ask for the **battleship new country name**. Validate that the length of the battleship’s country name must be **between 5 and 20 characters** **(inclusive)**.
* Ask for the **battleship new name**. Validate that the length of the battleship’s name must be **between 5 and 20 characters (inclusive)** and the name must **end with ‘ [first letter of company name]MS’**.
* Ask for the **battleship new total crew**. Validate that the battleship’s total crew must be **higher than 0**.
* Ask for the **battleship new availability**. Validate that the input only can be **either ‘y’ or ‘n’ (case insensitive)**.
* Ask for the **battleship new total troops**. Validate that the battleship’s total troops must be **higher than 0**.
* Finally, **update** the selected battleship’s data with the new battleship’s data.

Text

Description automatically generated

Figure 10. Update Battleship

1. **Exit**

Exit out of the program.

Text

Description automatically generated with medium confidence

Figure 11. Exit

**If you do not understand, please ask your assistant! Do not make your own assumption!**