Gebze Technical University

Computer Engineering

Object Oriented Analysis and Design

CSE443

Homework 1

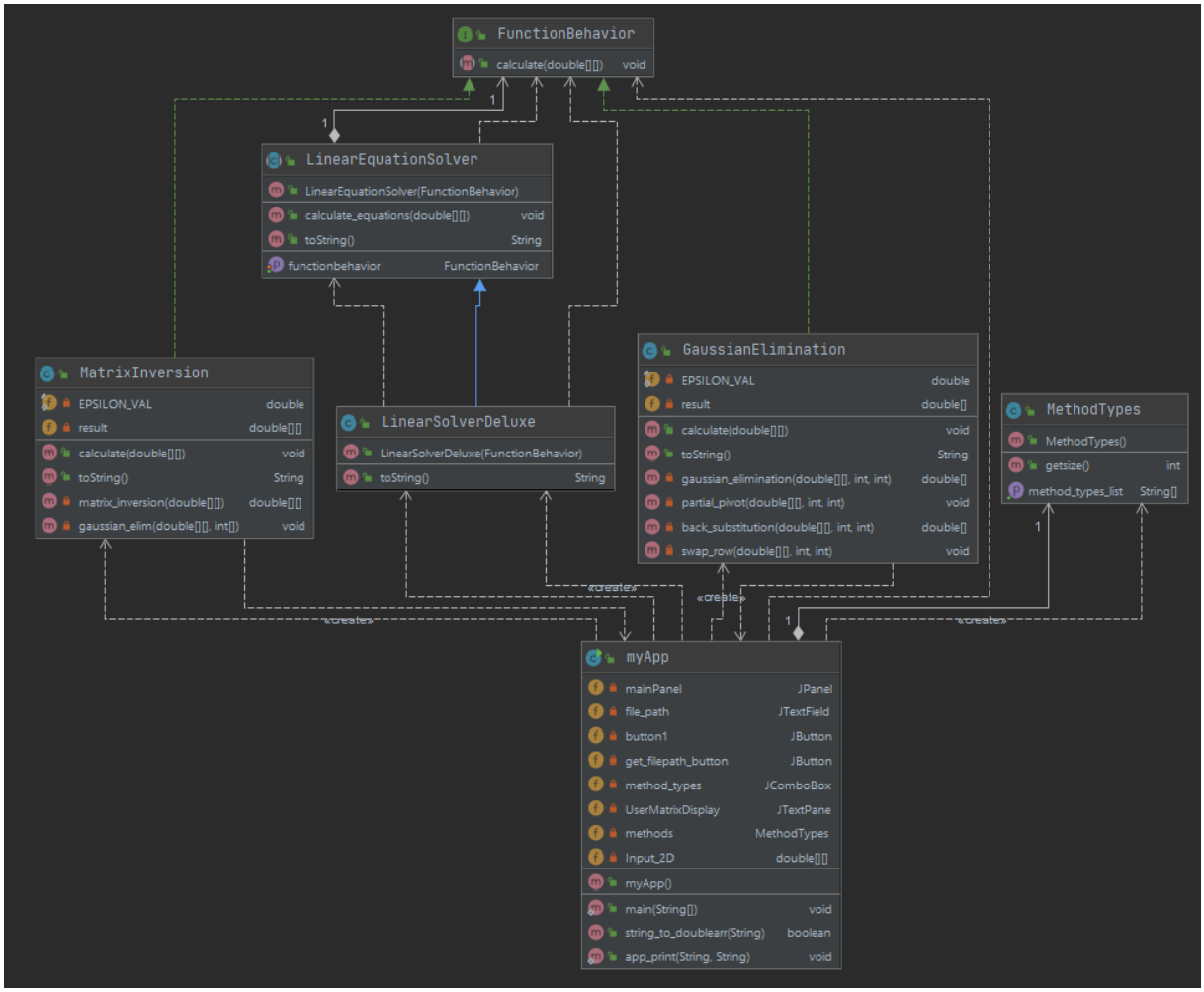
Can BEYAZNAR

161044038

1) Question 1 (LinearSolverDeluxe)

A) Solution

In this question, we are asked to write a program that can solve linear equations with 2 different methods. These methods are gaussian elimination and matrix inversion. The number of methods available in this program may increase. That's why we must choose a design pattern suitable for expansion. Also, there should be a dynamic transition between solving methods. The most suitable method for this problem is the strategy design pattern. Strategy design pattern defines a family of algorithms and wraps each one. And it makes them variable among themselves. If we need to apply this design pattern to our problem, the uml diagram is as follows:



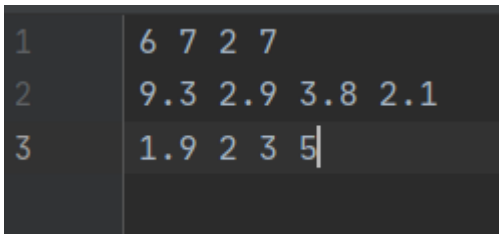
The GaussianElimination and MatrixInversion classes cover the methods we use. And these are implemented from the FunctionBehavior interface. FunctionBehavior class

only has calculate function. Thus, the encapsulation process is applied. The LinearEquationSolver class is used to call these encapsulated methods and is an abstract class. Methods in the LinearEquationSolver class inherit the LinearSolverDeluxe class. And so the customer can reach the solution of linear equations using this class. The reason for the LinearEquationSolver class is to lower maintenance costs in case of a different class such as LinearSolverDeluxe in the future. Finally, the MethodTypes class is used only to keep the names of methods (inside the String array) Using this class, we can learn the method chosen by the user more easily.

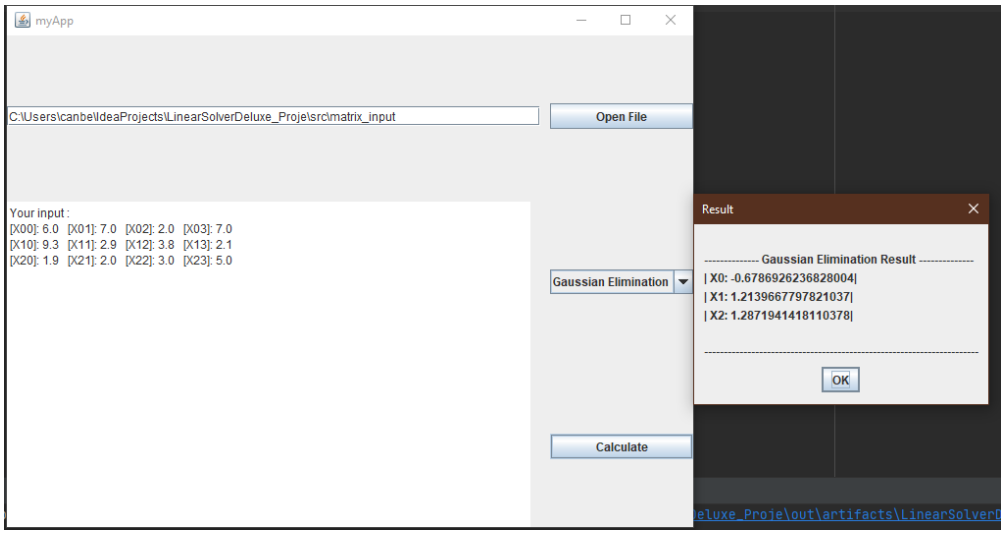
If another method comes in, such as Gaussian Elimination and Matrix Inversion methods (for example determinant calculation), we can easily add this new method. Write the name of the new method to the MethodTypes class. Using the FunctionBehavior interface, the determinantcalculation class is created and the necessary functions are implemented. And finally, this method is applied at the user's choice.

B) Output

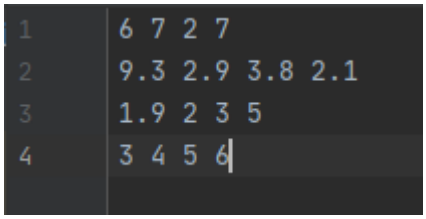
Matrix input for gaussian elimination:



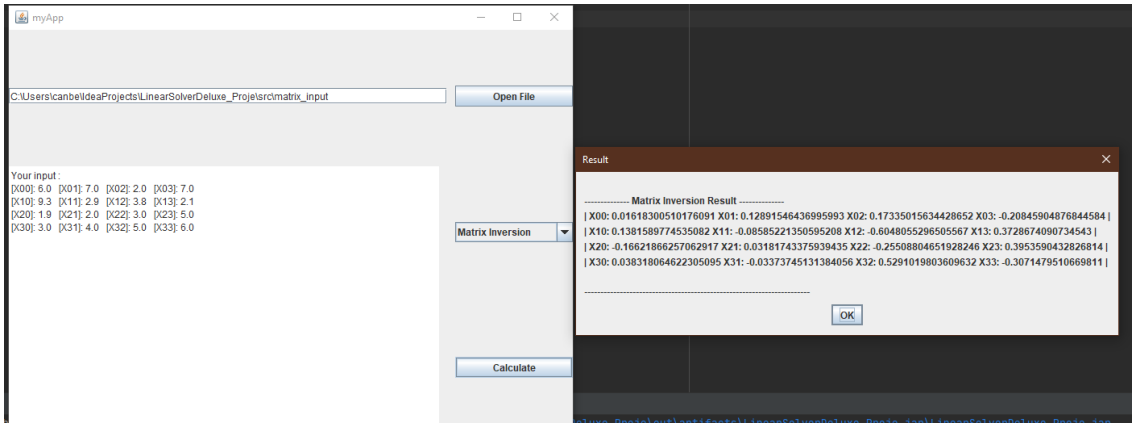
GUI App:



Matrix input for matrix inversion:

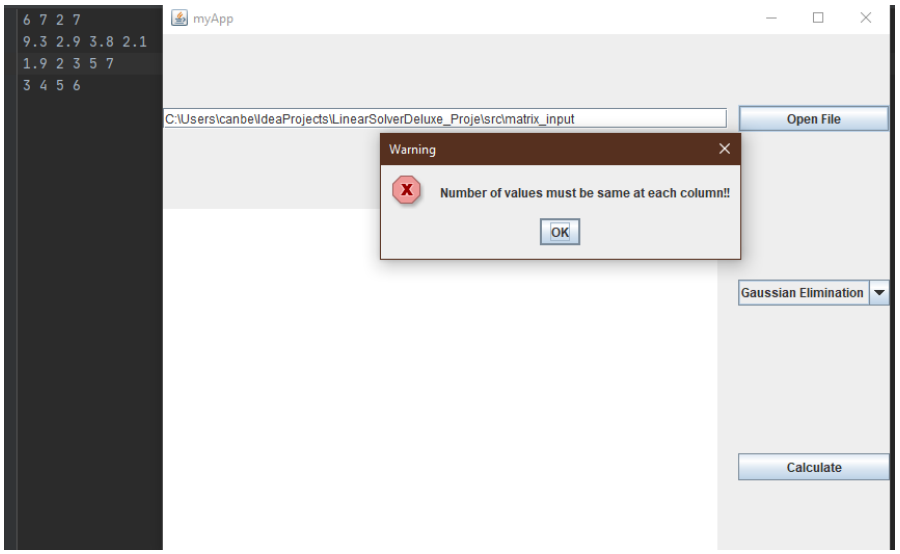


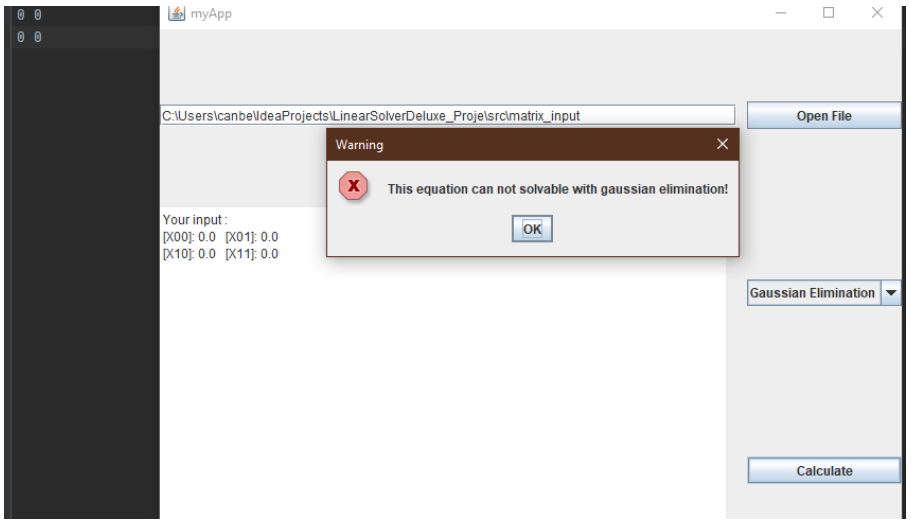
GUI App:

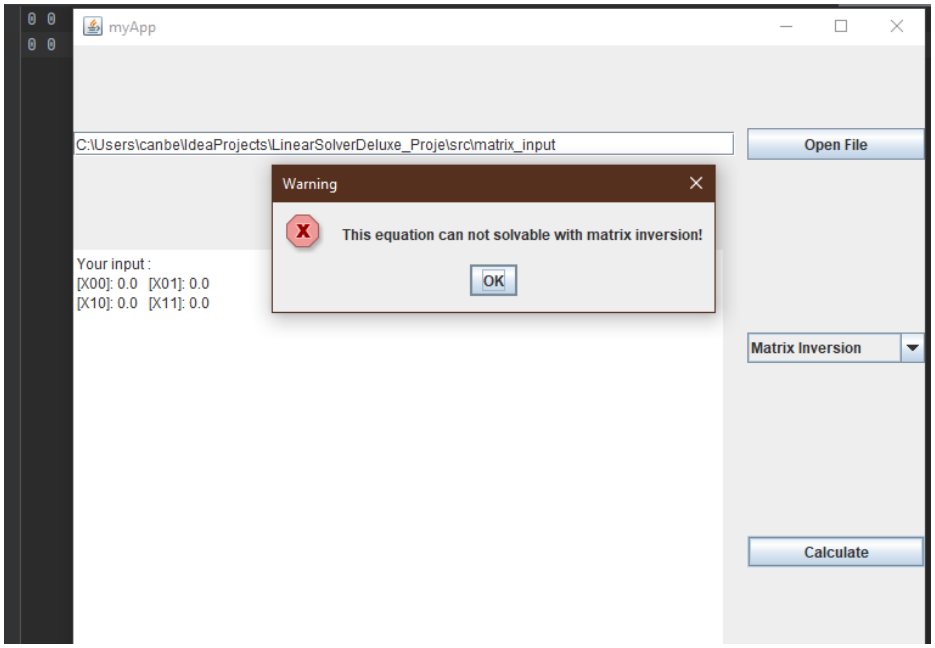


When testing the GUI application, first the path of the file is given and the file is opened. IMPORTANT rules that should be on file. There should be spaces between each number. Rows should be shown with newline. And in the application, the file input must be the absolute filepath of the file. for example: C: \ Users \ canbe \ IdeaProjects \ LinearSolverDeluxe_Project \ src \ matrix_input. Otherwise, the program will not work correctly.

Error Examples:



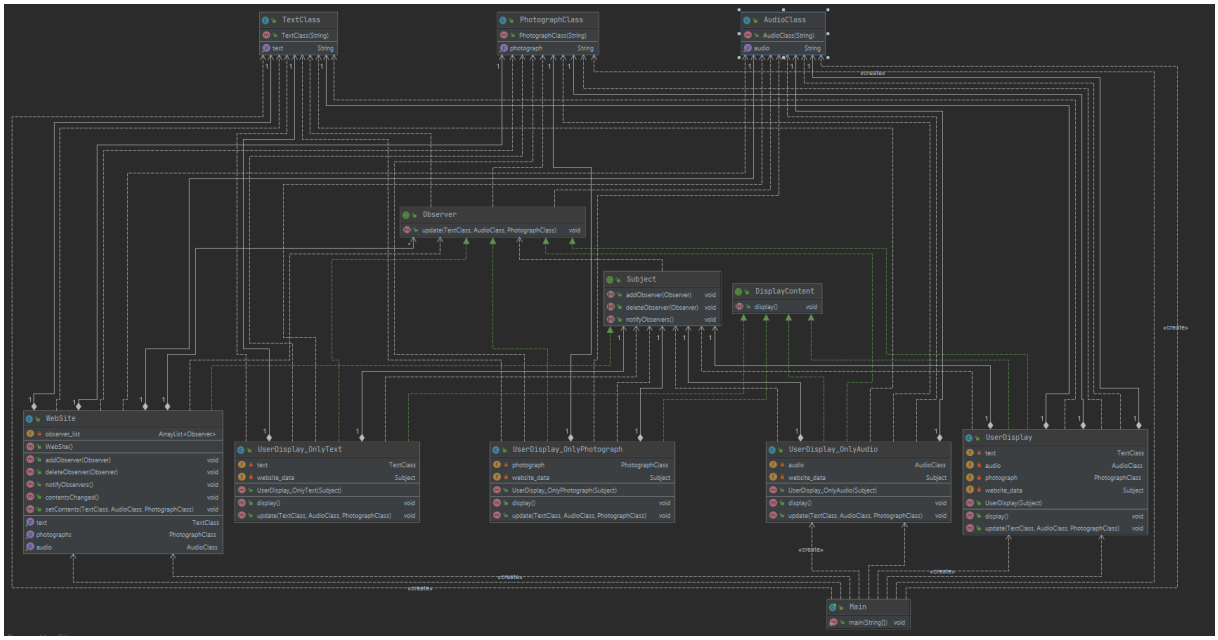




2) Question 2 (Website Problem)

A) Solution

In this problem, we need to show them notifications from users' favorite sites. And the content we will show you can be photos, sound and text. The most suitable method for this problem is observer design pattern. Observer design pattern defines a one-to-many relationship between a set of objects. Our subject is a website and we can assume our objects as users. So the UML diagram will look like this:



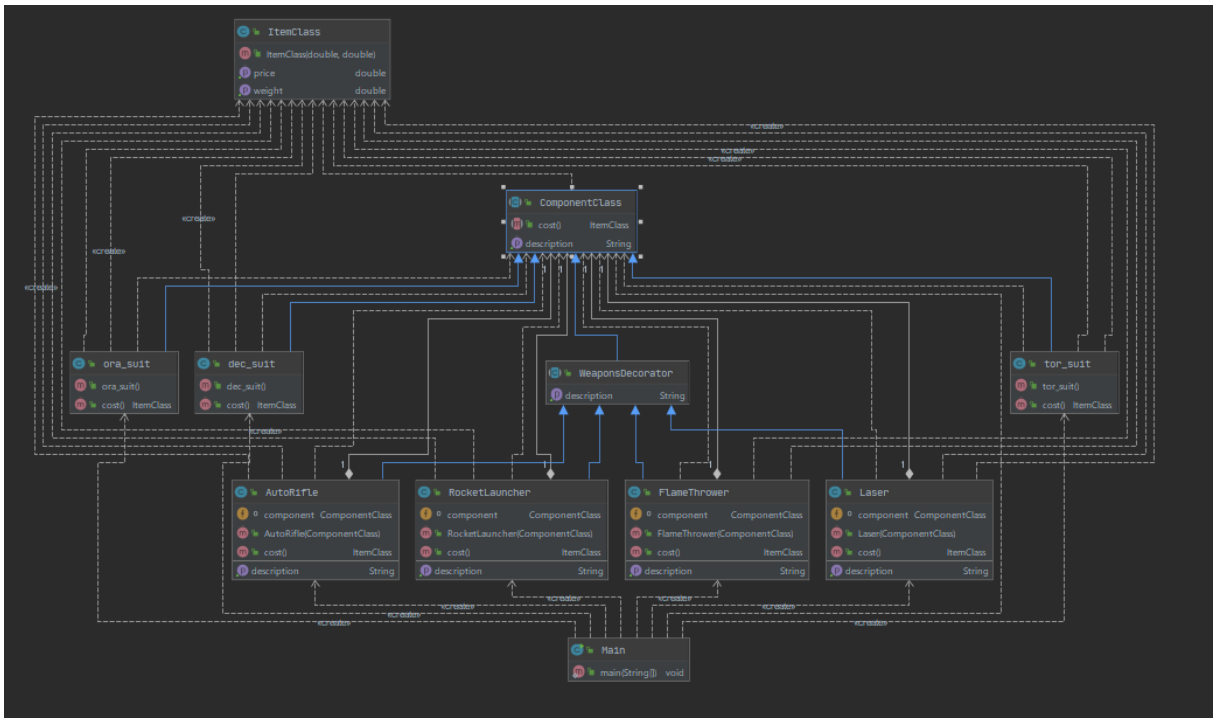
To implement this design pattern, I created 3 conten classes. TextClass, PhotographClass and AudioClass. These are used to convey the text, image and sound on the website to the users. Then I created the Subject interface. This interface will be used to register and delete users and send notifications to users. And I created a WebSite class implemented from it. The WebSite class contains methods from the Subject interface. And it keeps the list of users. Thus, when there is any change on the website, it sends a notification to users and enables users to access content. Observer interface is used to retrieve and update content from websites. UserDisplay, using the Observer interface, allows the user to access content from his favorite site. The display interface helps display the content in the UserDisplay class. The update method takes 3 parameters TextClass, PhotographClass and AudioClass. These classes are used to show the solution to the problem more clearly. These classes have only String parameters. The UserDisplay class can receive text, video and audio content from websites. Users who do not want to receive these 3 contents can use different User classes. For

example, the user who wants to only get audio content from his favorite website can use the UserUserDisplay_OnlyAudio class. With various user classes like this, the user can access the content they want to access. **Several of these combinations are implemented to avoid complexity in the UML diagram. To avoid confusion in the UML diagram, the content combinations have not been written any further.**

3) Question 3 (ZırhSan A.Ş)

A) Solution

To solve this problem, I used the Decorator design pattern. Decorator can dynamically add additional responsibilities to an object in run time. And it's one of the best ways to add new features to a classroom. First of all, I defined the ComponentClass class that will cover all components as abstract. In this class, descriptions of the weapons and armor purchased by the user will be accessed. And the total weight and total price of the products will be available. Then I collected the price and weight of each item in one class. The name of this class is ItemClass. Thus, we will be able to easily access the weight and price of the items. Then I produced the required armor classes. tor_suit, ora_suit and dec_suit. These extend from ComponentClass. And it can only be used once for a component. Then I defined an abstract class to define the required weapon classes. WeaponsDecorator class. Thus, users can buy more than one weapon. Then I defined the class of my weapons derived from the WeaponsDecorator class. Using this design pattern, he can add new weapons and armors. And the user can dynamically add multiple items to their inventory. The UML diagram of the structure I have described is as follows:



B) Output

I did not prepare any program here as in the first question. I have ready-made test cases in Main and print the results to the terminal.

Test code:

```
System.out.println("-o-o-o-o-o-o-o-o-o-o- Question 3 Test Case -o-o-o-o-o-o-o-o-o-o-");

System.out.println("First Component");
ComponentClass component = new dec_suit();
System.out.println(component.getDescription() + ", " + component.cost().getPrice() + "k TL, " + component.cost().getWeight() +" Kg");
System.out.println("-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-");


System.out.println("Second Component");
ComponentClass component2 = new ora_suit();
component2 = new FlameThrower(component2);
component2 = new AutoRifle(component2);
System.out.println(component2.getDescription() + ", " + component2.cost().getPrice() + "k TL, " + component2.cost().getWeight() +" Kg");
System.out.println("-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-");

System.out.println("Third Component");
ComponentClass component3 = new tor_suit();
component3 = new FlameThrower(component3);
component3 = new AutoRifle(component3);
component3 = new AutoRifle(component3);
component3 = new Laser(component3);
component3 = new RocketLauncher(component3);
System.out.println(component3.getDescription() + ", " + component3.cost().getPrice() + "k TL, " + component3.cost().getWeight() +" Kg");
```

Output:

```
-o-o-o-o-o-o-o-o-o-o- Question 3 Test Case -o-o-o-o-o-o-o-o-o-o-
First Component
Dec Suit, 500.0k TL, 25.0 Kg
-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-
Second Component
Ora Suit, FlameThrower, FlameThrower, 1580.0k TL, 33.5 Kg
-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-
Third Component
Tor Suit, FlameThrower, FlameThrower, FlameThrower, FlameThrower, FlameThrower, 5460.0k TL, 68.0 Kg
-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-
```