Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**ЗВІТ**

Про виконання лабораторної роботи **№ 6**

**«Реалізація функціоналу програми з індивідуального завдання»**

з дисципліни **«Прикладне програмування»**

**Лектор:**

доц. кафедри АСУ

Скорохода О.В.

**Виконав:**

студ. групи КН-203

Репей А.В.

**Прийняв:**

асис. кафедри АСУ

Мельник Р. В.

«\_\_\_» \_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема:** Реалізація функціоналу програми з індивідуального завдання.

**Мета:** Реалізувати функціонал програми з індивідуального завдання, звертаючи увагу на додаткові вимоги.

**Посилання на репозиторій Github**: https://github.com/temerep/labs\_ap/tree/main/knight\_ammunition

**Хід виконання лабораторної роботи:**

Створити консольну програму на мові програмування Java “Лицарі”.

Визначити ієрархію амуніції лицаря. Екіпірувати лицаря.

Підрахувати вартість. Здійснити сортування амуніції на основі ваги. Знайти елементи амуніції, що відповідають заданому діапазону параметрів ціни.

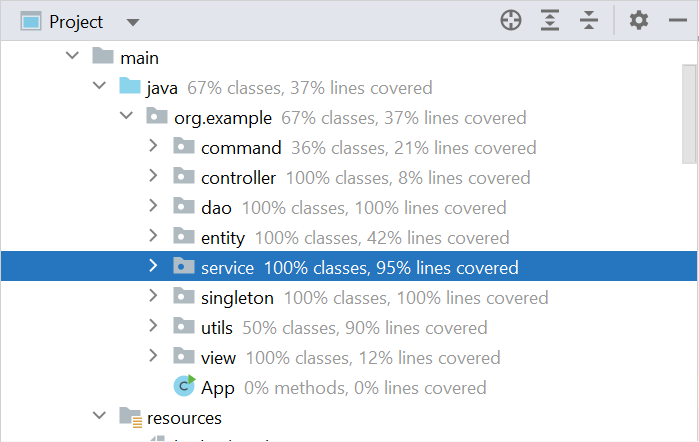
Окрім цього, програма повинна відповідати таким додатковим вимогам:  
• Використовувати можливості ООП: класи, успадкування, поліморфізм, інкапсуляцію.  
• Кожний клас повинен мати назву, яка повністю описує його суть, і інформативний склад.  
• Успадкування потрібно використовувати тільки тоді, коли воно має сенс.  
• При записі програми потрібно використовувати домовленості щодо оформлення коду java code convention.  
• Класи потрібно грамотно розкласти по пакетах.  
• Для збереження параметрів ініціалізації потрібно використовувати файли.  
• Повинне бути консольне меню для вибору різних дій, наприклад: «завантажити з файлу», «довідка», «вихід» тощо.

**Додаткове завдання**

Реалізуйте функціонал програми. Обов’язково зверніть увагу на додаткові вимоги до програми (описані вище).

**Хід виконання лабораторної роботи**

**Структура проекту**



Пакет command містить класи-команди, виконавця команд;

сontroller – контролери амуніції, dao – dao амуніції, entity – сутності амуніцій, а також фабрику для їх створення.

**Сутності**

***Амуніція:***

**package** org.example.entity;  
  
**public abstract class** Ammunition **extends** Entity **implements** Costable, Weightable, Durable {  
 **private** String **name**;  
 **private final double cost**;  
 **private final double weight**;  
  
 **private final double durability**;  
  
 **public** Ammunition(String name, **double** cost, **double** weight, **double** durability) {  
 **this**.**name** = name;  
 **this**.**cost** = cost;  
 **this**.**weight** = weight;  
 **this**.**durability** = durability;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 @Override  
 **public double** getCost() {  
 **return cost**;  
 }  
  
 @Override  
 **public double** getWeight() {  
 **return weight**;  
 }  
  
 @Override  
 **public double** getDurability() {  
 **return durability**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return new** StringBuilder()  
 .append(**this**.getClass().getSimpleName())  
 .append(**" "**)  
 .append(**name**)  
 .append(**" [id="**)  
 .append(**id**)  
 .append(**", cost="**)  
 .append(String.*format*(**"%.2f"**, **cost**))  
 .append(**", weight="**)  
 .append(String.*format*(**"%.2f"**, **weight**))  
 .append(**", "**)  
 .toString();  
 }  
}

***Нагрудник:***

**package** org.example.entity;  
  
**public class** ChestPlate **extends** Ammunition **implements** Protectable {  
 **private final double protection**;  
  
 **public** ChestPlate(String name, **double** cost, **double** weight, **double** durability, **double** protection) {  
 **super**(name, cost, weight, durability);  
 **this**.**protection** = protection;  
 }  
  
 @Override  
 **public double** getProtection() {  
 **return protection**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return super**.toString() +  
 **"protection="** + String.*format*(**"%.2f"**,**protection**) +  
 **']'**;  
 }  
}

***Шолом:***

**package** org.example.entity;  
  
**public class** Helmet **extends** Ammunition {  
 **private final** VisionLevel **visionLevel**;  
  
 **public** Helmet(String name, **double** cost, **double** weight, **double** durability, VisionLevel visionLevel) {  
 **super**(name, cost, weight, durability);  
 **this**.**visionLevel** = visionLevel;  
 }  
  
 **public** VisionLevel getVisionLevel() {  
 **return visionLevel**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return super**.toString() + **"vision level "** + **visionLevel** + **"]"**;  
 }  
}

***Щит:***

**package** org.example.entity;  
  
**public class** Shield **extends** Ammunition **implements** Protectable{  
 **private final double protection**;  
  
 **public** Shield(String name, **double** cost, **double** weight, **double** durability, **double** protection) {  
 **super**(name, cost, weight, durability);  
 **this**.**protection** = protection;  
 }  
  
 @Override  
 **public double** getProtection() {  
 **return protection**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return super**.toString() +  
 **"protection="** + String.*format*(**"%.2f"**,**protection**) +  
 **']'**;  
 }  
}

***Меч:***

**package** org.example.entity;  
  
**public class** Sword **extends** Ammunition **implements** Damageable{  
 **private final double damage**;  
  
 **public** Sword(String name, **double** cost, **double** weight, **double** durability, **double** damage) {  
 **super**(name, cost, weight, durability);  
 **this**.**damage** = damage;  
 }  
  
 @Override  
 **public double** getDamage() {  
 **return damage**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return super**.toString()+ **"damage="** + **damage** +  
 **']'**;  
 }  
}

Обробка даних із сховища, яке відіграє роль звичайного ліста:

***AbstractDao:***

**package** org.example.dao;  
  
**import** org.example.entity.Entity;  
  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** AbstractDao<T **extends** Entity> **implements** IDao<T> {  
 List<T> **storage**;  
  
 **public** AbstractDao(List<T> storage) {  
 **this**.**storage** = storage;  
 }  
  
 **public** AbstractDao() {  
 **storage** = **new** ArrayList<>();  
 }  
  
 @Override  
 **public** List<T> findAll() **throws** Exception {  
 **return new** ArrayList<>(**storage**);  
 }  
  
 @Override  
 **public** T findEntityById(**int** id) **throws** Exception {  
 **return storage**.get(id);  
 }  
  
 @Override  
 **public boolean** delete(T t) **throws** Exception {  
 **return storage**.remove(t);  
 }  
  
 @Override  
 **public boolean** delete(**int** id) **throws** Exception {  
 **storage**.remove(id);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** create(T t) **throws** Exception {  
 t.setId(**storage**.size());  
 **return storage**.add(t);  
 }  
}

Сервіс із реалізованою бізнес-логікою AmmunitionService:

**package** org.example.service;  
  
**import** org.example.dao.IDao;  
**import** org.example.entity.Ammunition;  
  
**import** java.util.ArrayList;  
**import** java.util.Comparator;  
**import** java.util.List;  
**import** java.util.function.Predicate;  
**import** java.util.stream.Collectors;  
  
**public class** AmmunitionService {  
 IDao<Ammunition> **dao**;  
  
 **public** AmmunitionService(IDao<Ammunition> dao) {  
 **this**.**dao** = dao;  
 }  
  
 **public** List<Ammunition> findAll() **throws** Exception {  
 **return dao**.findAll();  
 }  
  
 **public** Ammunition findEntityById(**int** id) **throws** Exception {  
 **return dao**.findEntityById(id);  
 }  
  
 **public boolean** delete(Ammunition ammunition) **throws** Exception {  
 **return dao**.delete(ammunition);  
 }  
  
 **public boolean** delete(**int** id) **throws** Exception {  
 **return dao**.delete(id);  
 }  
  
 **public boolean** create(Ammunition ammunition) **throws** Exception {  
 **return dao**.create(ammunition);  
 }  
  
 **public double** computeTotalEquippedAmmunitionCost() **throws** Exception {  
 List<Ammunition> all = **dao**.findAll();  
 **return** all.stream()  
 .map(Ammunition::getCost)  
 .reduce(0d, Double::*sum*);  
 }  
  
 **public** List<Ammunition> getAmmunitionListSortedByWeight() **throws** Exception {  
 List<Ammunition> all = **dao**.findAll();  
 all.sort(Comparator.*comparingDouble*(Ammunition::getWeight));  
 **return** all;  
 }  
  
 **public** List<Ammunition> getAmmunitionListInCostRange(**long** inclusiveStart, **long** inclusiveEnd) **throws** Exception {  
 List<Ammunition> all = **dao**.findAll();  
 Predicate<Ammunition> isInCostRange = x -> x.getCost() >= inclusiveStart  
 && x.getCost() <= inclusiveEnd;  
 **return** all.stream()  
 .filter(isInCostRange)  
 .collect(Collectors.*toCollection*(ArrayList::**new**));  
 }  
  
}

***Контролер AmmunitionController:***

**package** org.example.controller;  
  
**import** org.slf4j.LoggerFactory;  
**import** org.slf4j.Logger;  
**import** org.example.entity.Ammunition;  
**import** org.example.service.AmmunitionService;  
**import** org.example.utils.ReflectionUtils;  
**import** org.example.view.AmmunitionView;  
  
**import** java.util.List;  
  
**public class** AmmunitionController {  
 **private static final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(AmmunitionService.**class**.getName());  
 AmmunitionService **ammunitionService**;  
 AmmunitionView **ammunitionView**;  
  
 **public** AmmunitionController(AmmunitionService ammunitionService) {  
 **this**.**ammunitionService** = ammunitionService;  
 **ammunitionView** = **new** AmmunitionView(**this**, ammunitionService);  
 }  
  
 **public void** initView() {  
 **ammunitionView**.initView();  
 }  
  
 **public void** updateView() {  
 **ammunitionView**.updateView();  
 }  
  
  
 **public void** findAll() {  
 **try** {  
 **ammunitionView**.printAmmunition(**ammunitionService**.findAll());  
 } **catch** (Exception e) {  
  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
 **public void** findEntityById(**int** id) {  
 **try** {  
 Ammunition ammunitionById = **ammunitionService**.findEntityById(id);  
 **ammunitionView**.printAmmunitionById(id, ammunitionById);  
 } **catch** (java.lang.Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
 **public void** delete(Ammunition ammunition){  
 **try** {  
 **ammunitionService**.delete(ammunition);  
 **ammunitionView**.printDeleted(ammunition);  
 } **catch** (Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
 **public void** delete(**int** id) {  
 **try** {  
 Ammunition entityById = **ammunitionService**.findEntityById(id);  
 **ammunitionService**.delete(id);  
 **ammunitionView**.printDeleted(entityById);  
 } **catch** (java.lang.Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
 **public void** create(Ammunition ammunition) {  
 **try** {  
 **ammunitionService**.create( ammunition);  
 **ammunitionView**.printCreated(ammunition);  
 } **catch** (Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
  
 **public void** computeTotalEquippedAmmunitionCost() {  
 **try** {  
 **double** l = **ammunitionService**.computeTotalEquippedAmmunitionCost();  
 **ammunitionView**.printTotalEquippedAmmunitionCost(l);  
 } **catch** (java.lang.Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
 **public void** getAmmunitionListSortedByWeight() {  
 **try** {  
 List<Ammunition> ammunitionListSortedByWeight = **ammunitionService**.getAmmunitionListSortedByWeight();  
 **ammunitionView**.printAmmunitionSortedByWeight(ammunitionListSortedByWeight);  
 } **catch** (java.lang.Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
  
 **public void** getAmmunitionListInCostRange(**long** inclusiveStart, **long** inclusiveEnd) {  
 **try** {  
 List<Ammunition> ammunitionListInCostRange = **ammunitionService**.getAmmunitionListInCostRange(inclusiveStart, inclusiveEnd);  
 **ammunitionView**.printAmmunitionInCostRange(ammunitionListInCostRange, inclusiveStart,inclusiveEnd);  
 } **catch** (java.lang.Exception e) {  
 ***LOGGER***.warn(**"Input data for "**+ReflectionUtils.*getCurrentExecutingMethod*() + **" was incorrect"**);  
 *// throw new RuntimeException(e);* }  
 }  
}

***AmmunitionView:***

**package** org.example.view;  
  
**import** org.example.command.ICommand;  
**import** org.example.controller.AmmunitionController;  
**import** org.example.entity.Ammunition;  
**import** org.example.service.AmmunitionService;  
  
**import** java.util.List;  
  
**public class** AmmunitionView {  
 AmmunitionController **ammunitionController**;  
 AmmunitionService **ammunitionService**;  
  
 **public** AmmunitionView(AmmunitionController ammunitionController, AmmunitionService ammunitionService) {  
 **this**.**ammunitionController** = ammunitionController;  
 **this**.**ammunitionService** = ammunitionService;  
 *//initView();* }  
  
 **public void** initView() {  
 System.***out***.println(**"Today is good day to put something on..."**);  
 updateView();  
 }  
  
 **public void** updateView() {  
 StringBuilder stringBuilder = **new** StringBuilder()  
 .append(**"1 -> Print all ammunition\n"**)  
 .append(**"2 -> Delete ammunition by ID\n"**)  
 .append(**"3 -> Print ammunition by ID\n"**)  
 .append(**"4 -> Craft ammunition\n"**)  
 .append(**"5 -> Print total ammunition cost\n"**)  
 .append(**"6 -> Print all ammunition sorted by weight\n"**)  
 .append(**"7 -> Print all ammunition in cost range\n"**)

.append(**"0 -> Exit program\n"**);  
 System.***out***.println(stringBuilder);  
 }  
  
 **public void** printAmmunition(List<Ammunition> ammunitionList) {  
 System.***out***.println(**"All equipped ammunition: "** + ammunitionList);  
 }  
  
 **public void** executeCommand(ICommand command) **throws** Exception {  
 command.execute();  
 }  
  
 **public void** printDeleted(Ammunition ammunition) {  
 System.***out***.println(ammunition + **"is deleted"**);  
 }  
  
 **public void** printAmmunitionById(**int** id, Ammunition ammunitionById) {  
 System.***out***.println(**"Found "** + ammunitionById + **" by id="** + id);  
 }  
  
 **public void** printCreated(Ammunition ammunition) {  
 System.***out***.println(ammunition + **" is created"**);  
 }  
  
 **public void** printTotalEquippedAmmunitionCost(**double** cost) {  
 System.***out***.println(**"Total equipped ammunition cost = "** + cost);  
 }  
  
 **public void** printAmmunitionSortedByWeight(List<Ammunition> ammunitionListSortedByWeight) {  
 System.***out***.println(**"All ammunition sorted by weight: "** + ammunitionListSortedByWeight);  
 }  
  
 **public void** printAmmunitionInCostRange(List<Ammunition> ammunitionListInCostRange, **long** inclusiveStart, **long** inclusiveEnd) {  
 String stringBuilder = **"All ammunition in range ["** +  
 inclusiveStart +  
 **":"** +  
 inclusiveEnd +  
 **"] by cost value:\n"** +  
 ammunitionListInCostRange;  
 System.***out***.println(stringBuilder);  
 }  
}

**Реалізація патерну Команда**

Базовий інтерфейс IСommand:

**package** org.example.command;  
  
**public interface** ICommand {  
 **void** execute() **throws** Exception;  
}

Абстрактний клас AbstractCommand:

**package** org.example.command;  
  
  
  
**import** org.example.controller.AmmunitionController;  
**import** org.example.singleton.ScannerSingleton;  
**import** org.slf4j.LoggerFactory;  
**import** org.slf4j.Logger;  
**import** java.util.Scanner;  
  
  
**public abstract class** AbstractCommand **implements** ICommand {  
 **protected final static** Logger ***LOGGER*** = LoggerFactory.*getLogger*(AbstractCommand.**class**.getSimpleName());  
 **protected final** Scanner **scanner** = ScannerSingleton.*getInstance*().getScanner();  
 **protected** AmmunitionController **ammunitionController**;  
  
 **public** AbstractCommand(AmmunitionController ammunitionController) {  
 **this**.**ammunitionController** = ammunitionController;  
 }  
  
 @Override  
 **public void** execute() {  
 ***LOGGER***.info(**this**.getClass().getSimpleName() + **" is executed"**);  
 }  
}

Імплементації команд:

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
**import** org.example.entity.AmmunitionFactory;  
  
**public class** CraftAmmunitionCommand **extends** AbstractCommand {  
 AmmunitionFactory **ammunitionFactory**;  
  
 **public** CraftAmmunitionCommand(AmmunitionController ammunitionController, AmmunitionFactory ammunitionFactory) {  
 **super**(ammunitionController);  
 **this**.**ammunitionFactory** = ammunitionFactory;  
 }  
  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 **ammunitionController**.create(**ammunitionFactory**.createRandomAmmunition());  
 }  
}

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** DeleteAmmunitionCommandById **extends** AbstractCommand {  
  
 **public** DeleteAmmunitionCommandById(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 System.***out***.println(**"Enter ammunition ID to be deleted: "**);  
 **int** nextInt = **scanner**.nextInt();  
 **ammunitionController**.delete(nextInt);  
 }  
}

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** EmptyCommand **extends** AbstractCommand {  
  
 **public** EmptyCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 System.***out***.println(**"Command by this ID does not exist..."**);  
 }  
}

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** ExitProgramCommand **extends** AbstractCommand {  
  
  
 **public** ExitProgramCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 **scanner**.close();  
 System.*exit*(0);  
 }  
}

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** PrintAllAmmunitionInCostRangeCommand **extends** AbstractCommand {  
  
 **public** PrintAllAmmunitionInCostRangeCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 System.***out***.println(**"Enter inclusive start and end boundaries for ammunition cost to be printed: "**);  
 **int** start = **scanner**.nextInt();  
 **int** end = **scanner**.nextInt();  
 **ammunitionController**.getAmmunitionListInCostRange(start, end);  
 }  
}

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** PrintAllAmmunitionSortedByWeightCommand **extends** AbstractCommand {  
  
 **public** PrintAllAmmunitionSortedByWeightCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 **ammunitionController**.getAmmunitionListSortedByWeight();  
 }  
}

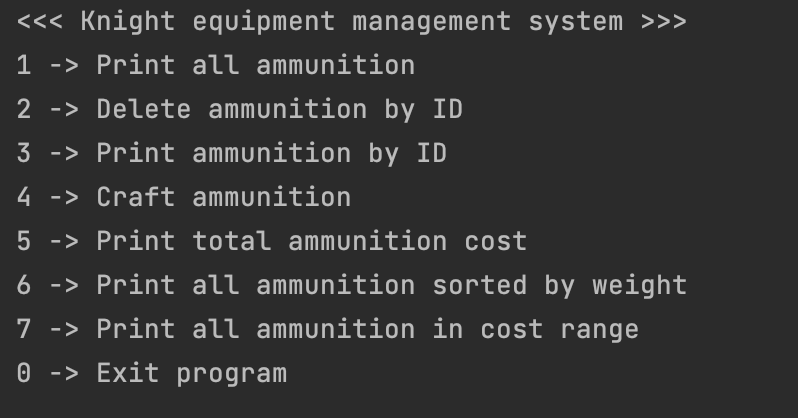
**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** PrintAmmunitionByIdCommand **extends** AbstractCommand {  
  
 **public** PrintAmmunitionByIdCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 System.***out***.println(**"Enter ammunition ID to be printed: "**);  
 **int** nextInt = **scanner**.nextInt();  
 **ammunitionController**.findEntityById(nextInt);  
 }  
}

**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** PrintTotalAmmunitionCommand **extends** AbstractCommand {  
  
 **public** PrintTotalAmmunitionCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 **ammunitionController**.findAll();  
 }  
}

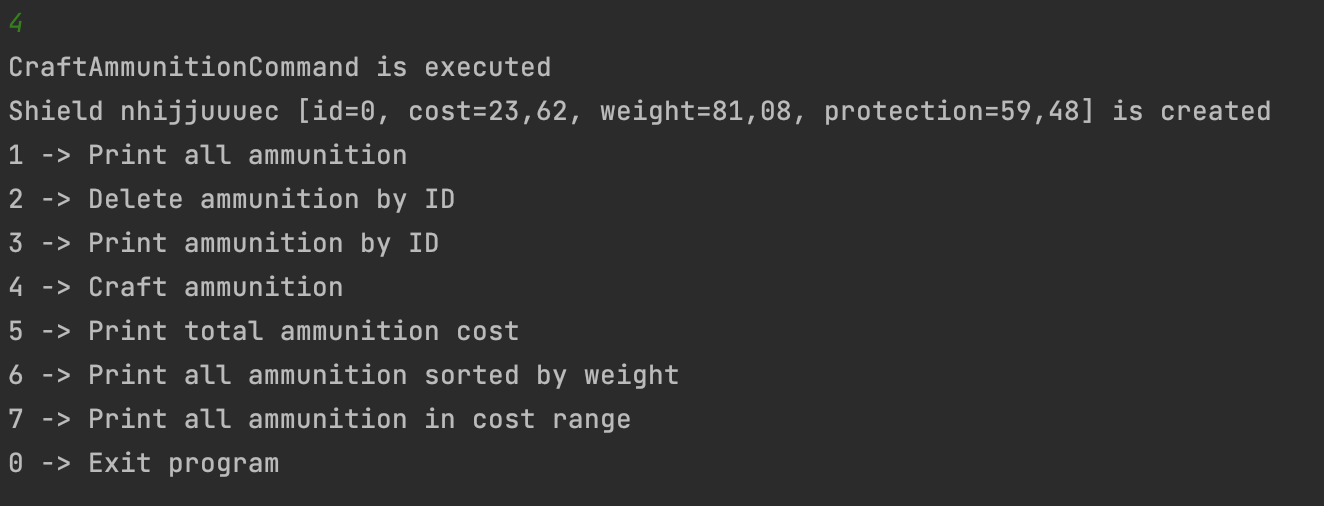
**package** org.example.command;  
  
**import** org.example.controller.AmmunitionController;  
  
**public class** PrintTotalAmmunitionCostCommand **extends** AbstractCommand {  
  
 **public** PrintTotalAmmunitionCostCommand(AmmunitionController ammunitionController) {  
 **super**(ammunitionController);  
 }  
  
 @Override  
 **public void** execute() {  
 **super**.execute();  
 **ammunitionController**.computeTotalEquippedAmmunitionCost();  
 }  
}

**Результат виконання програми**

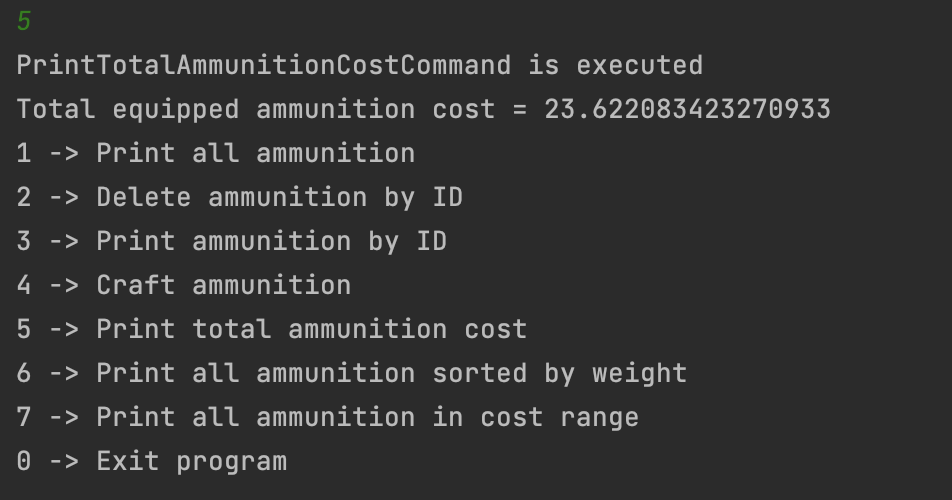
Запуск програми:

****

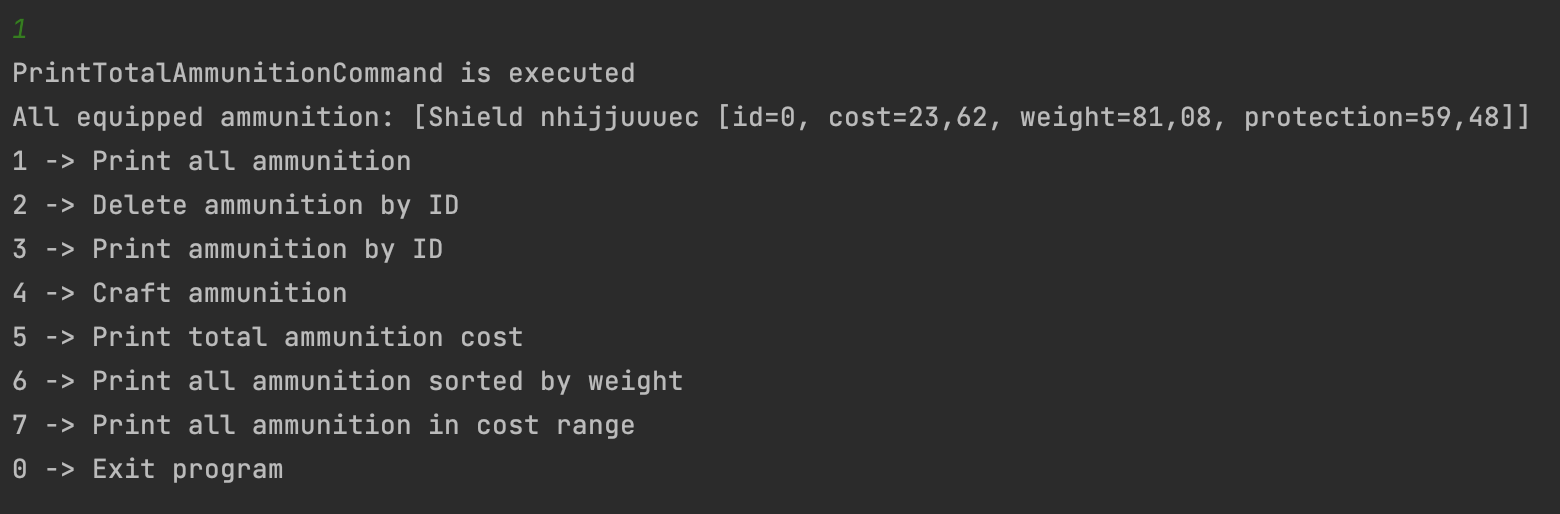
Крафт амуніції (щита) :

****

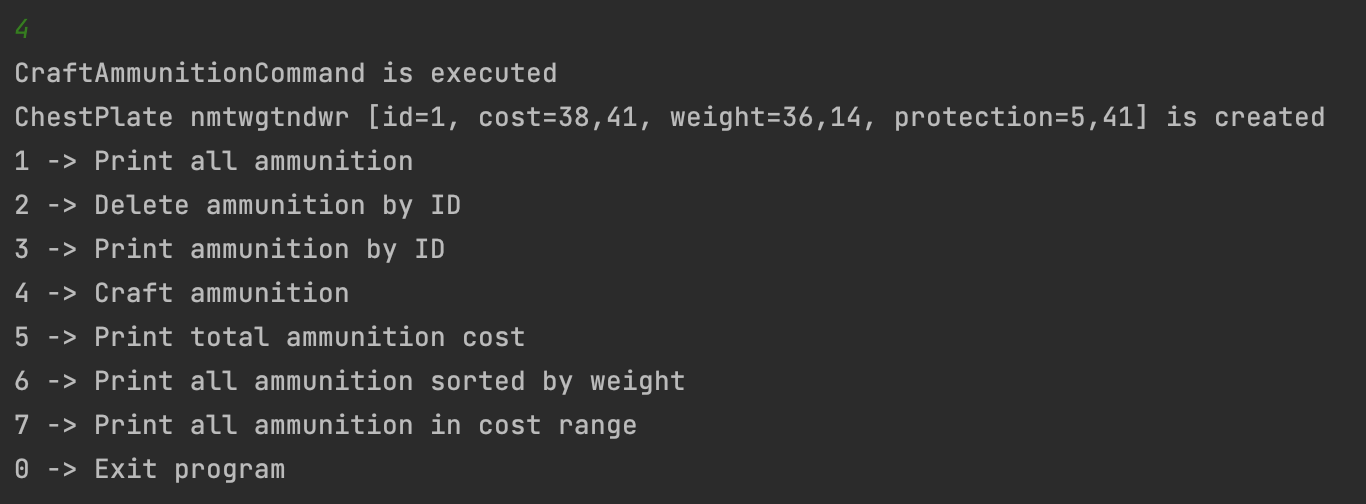
Вивід загальної вартості екіпірованої амуніції:

****

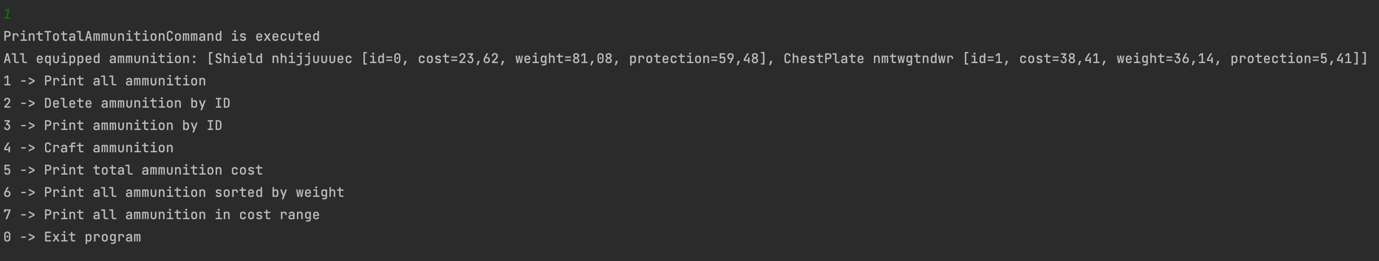
Вивід списку всіх амуніцій:

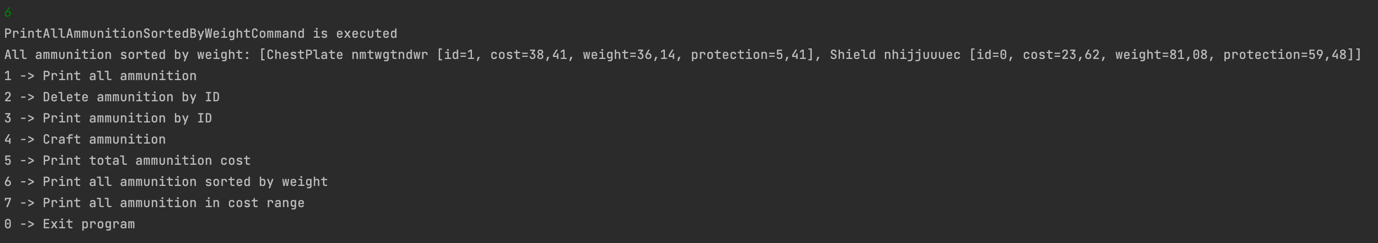
****

Крафт амуніції (нагрудника):

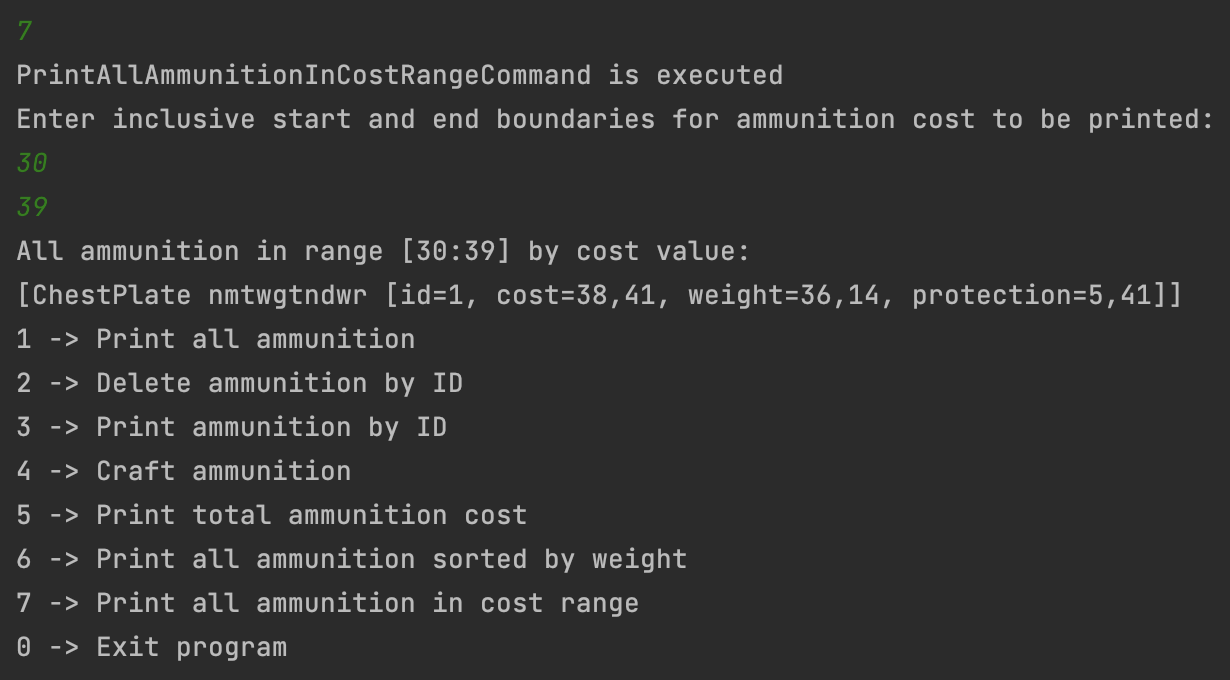
****

Вивід списку всіх амуніцій:

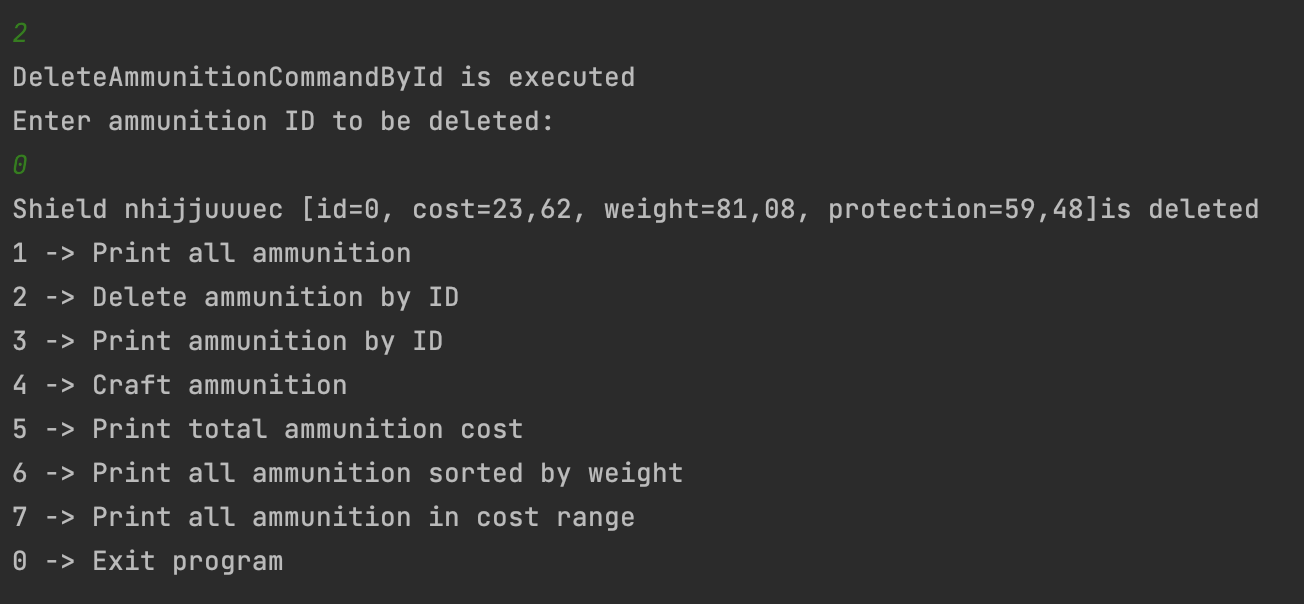
**** Вивід списку всіх амуніцій, відсортований за вагою:



Вивід списку амуніцій, які знаходяться в межах заданої вартості:

****

Видалення амуніції за ID:

****

**Висновок**

На лабораторній роботі я реалізував функціонал програми мого індивідуального завдання, дотримуючись додаткових вимог.