ASTANA IT UNIVERSITY

OOP endterm project. Character creation mode

Group members: IT-2207
Nurlybai Beksultan
Kashshafidin Sherkhan
Amanzhol Nurman

2023

Outline

1. Introduction

1.1. Description of the topic

The title of our project is “Character creation mode”. The topic name is explicitly related to video games, and this is not coincidence. The program is a part of some kind of RPG and role-playing games.

Character creation mode is a program that allows users to create and customize their own unique characters. This program offers a wide range of customization options, such as selecting classes, clothes, weapons, abilities of characters and even creating classes, clothes, weapons, abilities for characters. Obviously, characters have their own name and story. In addition, simple characteristics such as power, health, armor and damage per second depend on character’s class, clothing and weapon.

User can get, create, change and delete characters, classes, clothes, weapons and abilities from the console in the program. All changes are immediately stored in the database.

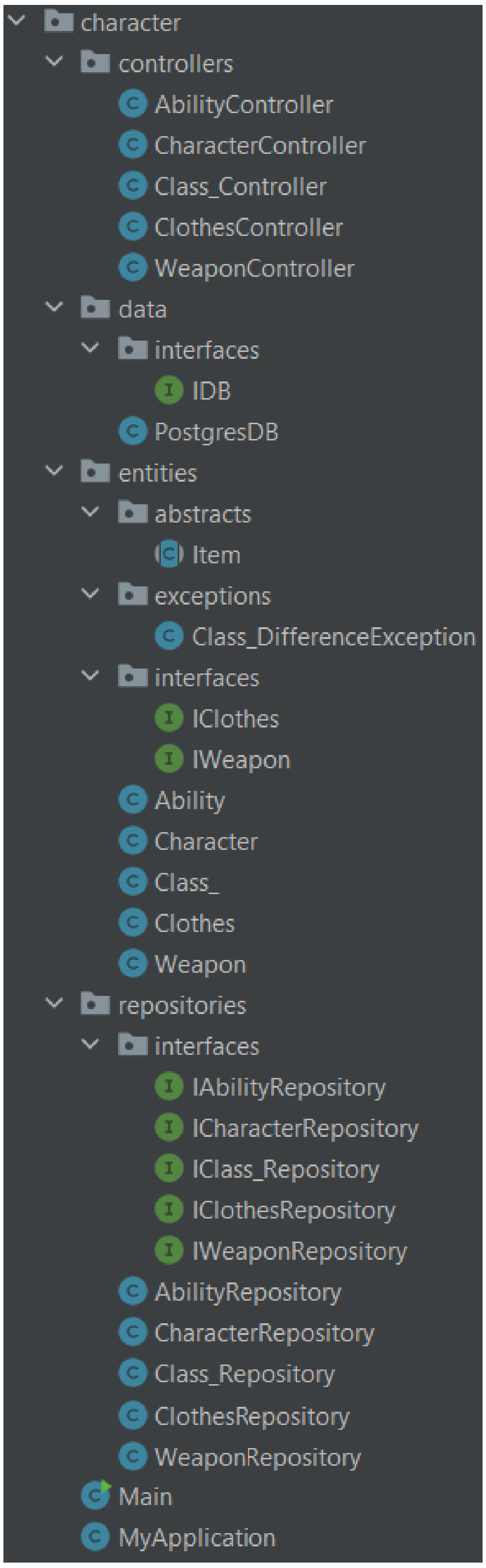
1.2. Team members and responsibilities

There are Nurlybai Beksultan, Kashshafidin Sherkhan and Amanzhol Nurman in the team. All of us from group IT-2207. In order to make a project, we separated the project into three pieces. Each responsibility of team members is follows:

- Amanzhol Nurman is responsible to code classes, interfaces, abstract class, their hierarchy and relations such as ‘Character’, ‘Class_’, ‘Weapon’, ‘Clothes’, ‘Ability’, ‘Item’, ‘IWeapon’ and ‘IClothes’.
- Kashshafidin Sherkhan is responsible to connect java classes with PostgreSQL database. So, a user or programmer can specify a request for obtaining, creating, changing or deleting data, the database response to this request.
- The idea and the main concept were developed by Nurlybai Beksultan. He is responsible to write an interface that is used to make a connection between user and program. It is a menu in console. In addition, responsibilities include writing documentation.

2. Structure of the project

2.1. Repository of the project



```
character
  controllers
    AbilityController
    CharacterController
    Class_Controller
    ClothesController
    WeaponController
  data
    interfaces
      IDB
    PostgresDB
  entities
    abstracts
      Item
    exceptions
      Class_DifferenceException
    interfaces
      IClothes
      IWeapon
    Ability
    Character
    Class_
    Clothes
    Weapon
  repositories
    interfaces
      IAbilityRepository
      ICharacterRepository
      IClass_Repository
      IClothesRepository
      IWeaponRepository
    AbilityRepository
    CharacterRepository
    Class_Repository
    ClothesRepository
    WeaponRepository
  Main
  MyApplication
```

Figure 1

Our program contains different kinds of files which separated by several packages. Main package, ‘character’ has java files that are ‘Main.java’ and ‘MyApplication.java’, and other packages such us ‘controllers’, ‘data’, ‘entities’ and ‘repositories’.

Our program runs from ‘Main.java’ file, but the menu that is used by a user to interact with the program in the console, is located in ‘MyAppication.java’ file.

Our main classes, which are used as entities in database tables, are in the ‘entities’ package. In addition, there are abstract class, interfaces and exception class in the package.

‘data’ package contains files that connect the program with database. In our case, database management system is PostgreSQL.

In ‘repositories’ package, there are classes for each entity class which contain methods to send query the database.

‘controller’ repository has classes that manage data that is passed from a programmer or a user to repository classes.

All files are connected and interact with each other. So, they form a whole program.

2.2 Structure of classes

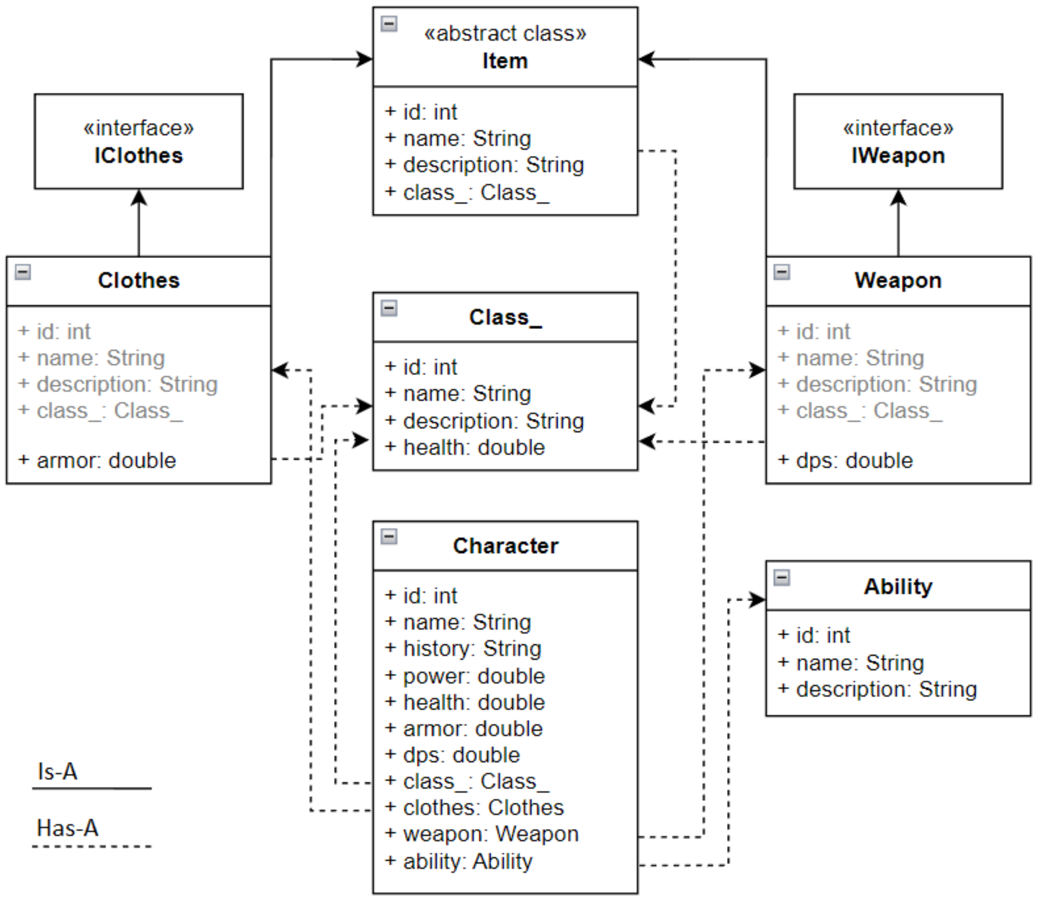


Figure 2

2.3. ERD

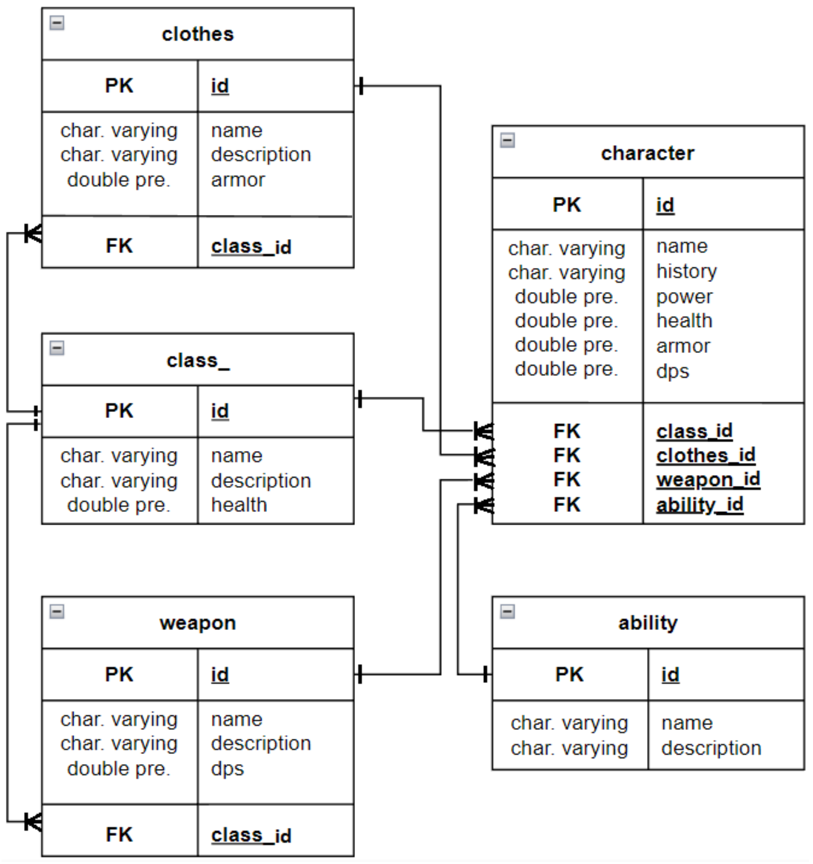


Figure 3

3. The source codes

3.1 Entity classes

In the package 'entities', there are such classes, abstract class, interfaces and exception class as Character, Class_, Clothes, Weapon, Ability, Item, IClothes, IWeapon, Class_DifferenceException (Figure 1), and their relationship is shown in figure 2. So, all class implementations are shown below. Some code snippets exist in multiple classes. Consequently, not whole code will be represented.

```
public abstract class Item {
    private int id;
    private String name;
    private String description;
    private Class_ class_;

    public abstract void describe();

    public void setId(int id) { this.id = id; }
    public void setName(String name) { this.name = name; }
    public void setDescription(String description) { this.description =
description; }
    public void setClass_(Class_ class_) { this.class_ = class_; }

    public int getId() { return id; }
    public String getName() { return name; }
    public String getDescription() { return description; }
    public Class_ getClass_() { return class_; }
}
```

```
public class Clothes extends Item implements IClothes {
    private double armor;

    public void setArmor(double armor) { this.armor = armor; }
    public double getArmor() { return armor; }

    public Clothes(String name, String description, Class_ class_, double
armor){
        setName(name);
        setDescription(description);
        setClass_(class_);
        setArmor(armor);
    }
    public Clothes(int id, String name, String description, Class_ class_,
double armor){
        this(name, description, class_, armor);
        setId(id);
    }

    @Override
    public void describe(){
        System.out.printf("\nId: %d\nName: %s\nClass: %s\nArmor:
%.0f\nDescription: %s\n\n",
                getId(), getName(), getClass_().getName(), getArmor(),
getDescription());
    }
    @Override
    public void takeDamage(){ System.out.printf("%s absorbed %.0f% of
damage.\n", getName(), getArmor()/10); }
    @Override
    public void dodge(){ System.out.println("You dodged the attack."); }

    @Override
    public String toString(){ return "clothes(" + getName() + ")"; }
}
```

```java
public class Character {
    private int id;
    private String name;
    private String history;

    public void setId(int id) { this.id = id; }
    public void setName(String name) { this.name = name; }
    public void setHistory(String history) { this.history = history; }

    private double health = 100;
    private double armor = 0;
    private double dps = 10;
    private double power = 250;

    private void setHealth(Class_ class_) { this.health = 100 +
class_.getHealth(); }
    private void setArmor(Clothes clothes) { this.armor = clothes.getArmor(); }
    private void setDps(Weapon weapon) { this.dps = weapon.getDps(); }
    private void setPower(){ this.power = getArmor() + getHealth()*2 +
getDps()*5; }

    private Class_ class_;
    private Ability ability;
    private Clothes clothes;
    private Weapon weapon;

    public Character(String name, String history){
        setName(name);
        setHistory(history);
    }

    public Character(int id, String name, String history){
        this(name, history);
        setId(id);
    }
    public Character(String name, String history, Class_ class_, Ability
ability, Clothes clothes, Weapon weapon){
        this(name, history);
        this.class_ = class_;
        this.ability = ability;
        this.clothes = clothes;
        this.weapon = weapon;

        setHealth(class_);
        setArmor(clothes);
        setDps(weapon);
        setPower();
    }
    public Character(int id, String name, String history, Class_ class_, Ability
ability, Clothes clothes, Weapon weapon){
        this(name, history, class_, ability, clothes, weapon);
        setId(id);
    }

    public void setClass_(Class_ class_) {
        if (class_ != this.class_){
            setHealth(class_);
            this.armor = 0;
            this.dps = 10;
            setPower();

            this.class_ = class_;
            this.clothes = null;
            this.weapon = null;
        }
    }
    public void setAbility(Ability ability) { this.ability = ability; }

    private boolean checkItem(Item item){
        try {
            if (this.getClass_() == null) setClass_(item.getClass_());
            if (item.getClass_() != this.getClass_())
                throw new Class_DifferenceException("The %s has different
class_!".formatted(item));
            return true;
        } catch (Class_DifferenceException diffEx) {
            diffEx.printStackTrace();
            System.exit(-1);
        }
        return false;
    }
    public void setClothes(Clothes clothes) {
        if (checkItem(clothes)){
            setArmor(clothes);
            setPower();
            this.clothes = clothes;
        }
    }
    public void setWeapon(Weapon weapon) {
        if (checkItem(weapon)){
            setDps(weapon);
            setPower();
            this.weapon = weapon;
        }
    }

    public void removeClass_(){
        if (getClass_() != null){
            this.health = 100;
            this.armor = 0;
            this.dps = 10;
            this.power = 250;

            this.class_ = null;
            this.clothes = null;
            this.weapon = null;
        }
    }
    public void removeAbility(){ if (getAbility() != null){ this.ability = null;
} }
    public void removeClothes(){
        if (getClothes() != null){
            this.armor = 0;
            setPower();
            this.clothes = null;
        }
    }
    public void removeWeapon(){
        if (getWeapon() != null){
            this.dps = 10;
            setPower();
            this.weapon = null;
        }
    }

    public int getId() { return id; }
    public String getName() { return name; }
    public String getHistory() { return history; }
    public double getHealth() { return health; }
    public double getArmor() { return armor; }
    public double getDps() { return dps; }
    public double getPower() { return power; }
    public Class_ getClass_() { return class_; }
    public Ability getAbility() { return ability; }
    public Clothes getClothes() { return clothes; }
    public Weapon getWeapon() { return weapon; }

    public boolean fight(Character enemy){
        return this.power >= enemy.power;
    }

    public String getStat(){
        return ("Id: %d\nName: %s\nHealth: %.0f\nPower: %.0f\nArmor: %.0f\nDPS: 
%.0f\n" +
                "Class: %s\nAbility: %s\nClothes: %s\nWeapon: %s\nHistory: 
%s\n").formatted(
                getId(), getName(), getHealth(), getPower(), getArmor(), 
getDps(),
                (getClass_()==null)?"null":getClass_().getName(),
                (getAbility()==null)?"null":getAbility().getName(),
                (getClothes()==null)?"null":getClothes().getName(),
                (getWeapon()==null)?"null":getWeapon().getName(), getHistory());
    }
}
```

3.2 Database connection

In order to connect database to the program, there are a file called 'PostgreDB' in the package 'data'.

```
public class PostgresDB implements IDB {
    @Override
    public Connection getConnection() throws SQLException, 
ClassNotFoundException {
        String connectionURL = 
"jdbc:postgresql://localhost:5432/character_creation_part";
        try {
            Class.forName("org.postgresql.Driver");
            Connection con = DriverManager.getConnection(connectionURL, 
"postgres", "admin");
            return con;
        } catch (Exception ex) {
            System.out.println(ex);
            return null;
        }
    }
}
```

Then, in order to send a query and get a response from database, there are classes for each entity classes such as CharacterRepository, Class_Repository, ClothesRepository, WeaponRepository and AbilityRepository. Each of them contains five similar methods that get a record, get all records, create a record, change a record and delete a record. So, there will be only one class to prevent information overload.

```
public class WeaponRepository implements IWeaponRepository {
    private final IDB db;
    public WeaponRepository(IDB db){
        this.db = db;
    }

    @Override
    public boolean createWeapon(Weapon weapon){
        Connection con = null;
        try {
            con = db.getConnection();
            String sql = "INSERT INTO weapon (name, description, dps, class_id) 
VALUES (?, ?, ?, ?);";
            PreparedStatement st = con.prepareStatement(sql);

            st.setString(1, weapon.getName());
```

```java
            st.setString(2, weapon.getDescription());
            st.setDouble(3, weapon.getDps());
            st.setInt(4, weapon.getClass_().getId());

            st.execute();
            return true;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        } finally {
            try {
                con.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return false;
    }

    @Override
    public Weapon getWeapon(int id){
        Connection con = null;
        try {
            con = db.getConnection();
            String sql = "Select id, name, description, dps, class_id FROM 
weapon WHERE id = ?";
            PreparedStatement st = con.prepareStatement(sql);
            st.setInt(1, id);
            ResultSet rs = st.executeQuery();
            if (rs.next()){
                String sql2 = "Select id, name, description, health FROM class_ 
WHERE id = ?";
                PreparedStatement st2 = con.prepareStatement(sql2);
                st2.setInt(1, rs.getInt("class_id"));
                ResultSet rs2 = st2.executeQuery();
                Class_ class_ = null;
                if (rs2.next()) {
                    class_ = new Class_(
                            rs2.getInt("id"),
                            rs2.getString("name"),
                            rs2.getString("description"),
                            rs2.getDouble("health"));
                }

                Weapon weapon = new Weapon(
                        id,
                        rs.getString("name"),
                        rs.getString("description"),
                        class_,
                        rs.getDouble("dps"));
                return weapon;
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        } finally {
            try {
                con.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return null;
    }
    @Override
    public boolean changeWeapon(int id, String name, String description, Class_
class_, double dps){
        Connection con = null;
        try {
            con = db.getConnection();
            String sql = "UPDATE weapon SET ";
            boolean temp = false;
            if (name != null){
                sql = sql.concat("name = " + "\'" + name + "\'");
                temp = true; }
            if (description != null){
                if (temp){ sql = sql.concat(", description = " + "\'" +
description + "\'");
                } else { sql = sql.concat("description = " + "\'" + description
+ "\'"); }
                temp = true;
            }
            if (dps != 0){
                if (temp){ sql = sql.concat(", dps = " + dps);
                } else { sql = sql.concat("dps = " + dps); }
                temp = true;
            }
            if (class_ != null){
                if(temp){ sql = sql.concat(", class_id = " + class_.getId());
                } else { sql = sql.concat("class_id = " + class_.getId()); }
            }

            sql = sql.concat(" WHERE id = " + id);
            Statement st = con.createStatement();
            st.executeUpdate(sql);
            return true;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        } finally {
            try {
                con.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return false;
    }

    @Override
    public boolean deleteWeapon(int id){
        Connection con = null;
        try {
            con = db.getConnection();
            String sql = "DELETE FROM weapon WHERE id = " + id;
            Statement st = con.createStatement();
            st.executeUpdate(sql);
            return true;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        } finally {
            try {
                con.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return false;
    }

    @Override
    public List<Weapon> getAllWeapon(){
        Connection con = null;
        try {
            con = db.getConnection();
            Statement st = con.createStatement();

            ResultSet rs = st.executeQuery("Select *  FROM weapon");
            List<Weapon> weapons = new LinkedList<>();

            IClass_Repository classRepo = new Class_Repository(db);
            List<Class_> class_s = classRepo.getAllClass_();

            while (rs.next()){
                Class_ class_ = null;
                for (int i=0; i< class_s.size(); i++){
                    if (class_s.get(i).getId() == rs.getInt("class_id")){
                        class_ = class_s.get(i);
                    }
                }
                Weapon weapon = new Weapon(
                        rs.getInt("id"),
                        rs.getString("name"),
                        rs.getString("description"),
                        class_,
                        rs.getDouble("dps"));
                weapons.add(weapon);
            }
            return weapons;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        } finally {
            try {
                con.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return null;
    }
}
```

To control how data flow from files to database, there are classes for each repository classes such as CharacterController, Class_Controller, ClothesClotroller, WeaponController and AbilityController. One of them is shown below.

```
public class WeaponController {
    private final IWeaponRepository repo;
    public WeaponController (IWeaponRepository repo) { this.repo = repo; }

    public String createWeapon (String name, String description, Class_ class_,
double dps){
        Weapon weapon = new Weapon (name, description, class_, dps);

        boolean created = repo.createWeapon(weapon);

        return (created) ? "Weapon was created!" : "Weapon creation was
failed!";
    }
    public Weapon getWeapon(int id){ return repo.getWeapon(id); }

    public String changeWeapon(int id, String name, String description, Class_
class_, double dps){
        boolean done = repo.changeWeapon(id, name, description, class_, dps);
        return (done) ? "Weapon was changed!" : "Weapon was were failed!";
    }

    public String deleteWeapon(int id){
        boolean done = repo.deleteWeapon(id);
        return (done) ? "Weapon was deleted!" : "Weapon removal was failed!";
    }

    public List<Weapon> getAllWeapon() {
        return repo.getAllWeapon();
    }
}
```

3.3 User interface

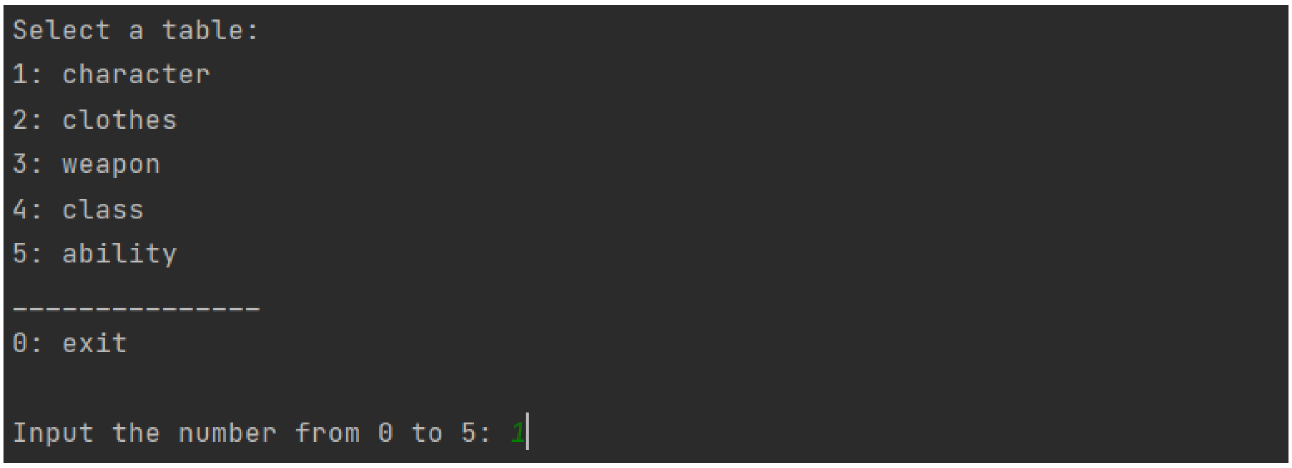
To provide opportunity for users to interact with the program, there is a file called ‘MyApplication.java’ which has the menu code that runs in the console for users. But before that, the program would be run in the file called ‘Main.java’.

```
public class Main {
    public static void main(String[] args){

        IDB db = new PostgresDB();
        ICharacterRepository characterRepo = new CharacterRepository(db);
        IClass_Repository classRepo = new Class_Repository(db);
        IAbilityRepository abilityRepo = new AbilityRepository(db);
        IClothesRepository clothesRepo = new ClothesRepository(db);
        IWeaponRepository weaponRepo = new WeaponRepository(db);

        CharacterController characterController = new
CharacterController(characterRepo);
        Class_Controller classController = new Class_Controller(classRepo);
        AbilityController abilityController = new
AbilityController(abilityRepo);
        ClothesController clothesController = new
ClothesController(clothesRepo);
        WeaponController weaponController = new WeaponController(weaponRepo);

        MyApplication myApp = new MyApplication(characterController,
classController,
                abilityController, clothesController, weaponController);

        myApp.start();
    }
}
```

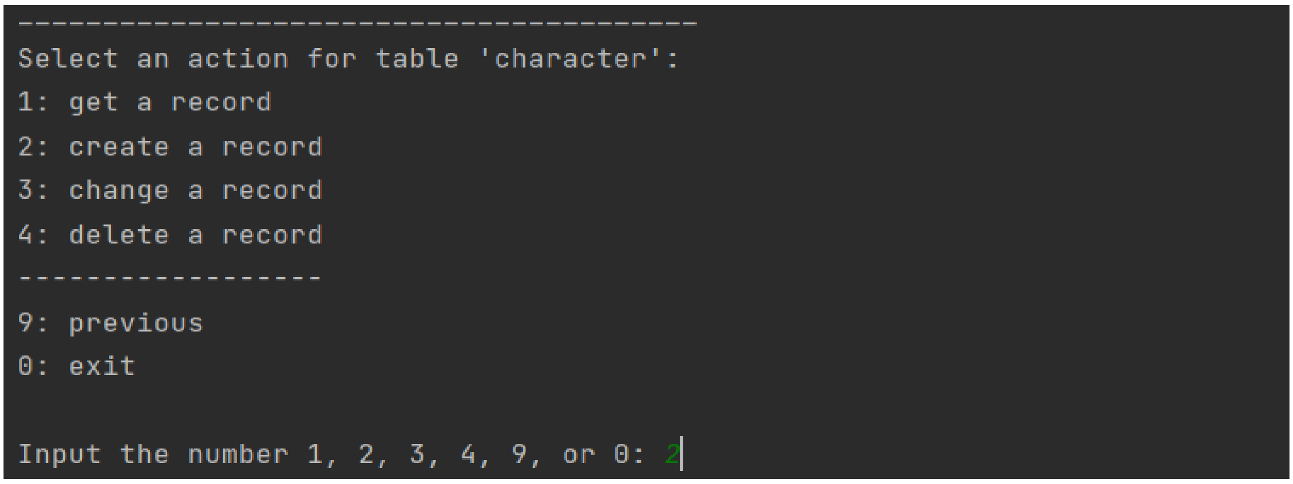
```
public class MyApplication {
    private final CharacterController characterController;
    private final Class_Controller classController;
    private final AbilityController abilityController;
    private final ClothesController clothesController;
    private final WeaponController weaponController;
    private final Scanner scanner;

    public MyApplication(CharacterController characterController,
Class_Controller classController,
                         AbilityController abilityController, ClothesController
clothesController,
                         WeaponController weaponController){
        this.characterController = characterController;
        this.classController = classController;
        this.abilityController = abilityController;
```

```java
        this.clothesController = clothesController;
        this.weaponController = weaponController;
        this.scanner = new Scanner(System.in);
    }

    public void start(){
        System.out.println("Welcome to our application!");
        while (true){
            System.out.println("""
                \nSelect a table:
                1: character
                2: clothes
                3: weapon
                4: class
                5: ability
                _______________
                0: exit\n""");
            outer: while (true){
                System.out.print("Input the number from 0 to 5: ");
                int tbIndex = scanner.nextInt();
                switch (tbIndex){
                    case 1:
                        separationLine();
                        characterEdit();
                        break outer;
                    case 2:
                        separationLine();
                        clothesEdit();
                        break outer;
                    case 3:
                        separationLine();
                        weaponEdit();
                        break outer;
                    case 4:
                        separationLine();
                        classEdit();
                        break outer;
                    case 5:
                        separationLine();
                        abilityEdit();
                        break outer;
                    case 0:
                        System.exit(0);
                        break outer;
                    default:
                        System.out.println("the input must be between 0 and
5.");
                        break;
                }
            }
        }
    }

                                              …

    public void weaponEdit(){
        outer2: while (true){
            System.out.println("""
                Select an action for table 'weapon':
                1: get a record
                2: create a record
                3: change a record
                4: delete a record
                ------------------
                9: previous
                0: exit\n""");
            outer: while (true){
                System.out.print("Input the number 1, 2, 3, 4, 9, or 0: ");
                int index = scanner.nextInt();
                int id, temp;
                double dps;
                String name, description, message;
                Class_ class_;
                switch (index){
                    case 1:

System.out.println(toTable(weaponController.getAllWeapon()));
                        System.out.print("Input id of the weapon: ");
                        id = scanner.nextInt();
                        Weapon weapon = weaponController.getWeapon(id);
                        if (weapon == null){
                            System.out.println("The weapon was not found.");
                        } else {
                            weapon.describe();
                            System.out.print("Input any letter to continue: ");
                            scanner.next();
                        }
                        break outer;
                    case 2:

System.out.println(toTable(weaponController.getAllWeapon()));
                        System.out.print("Input name of the weapon: ");
                        scanner.nextLine();
                        name = scanner.nextLine();

                        System.out.print("Input description of the weapon: ");
                        description = scanner.nextLine();

                        System.out.print("Input dps of the weapon: ");
                        dps = scanner.nextDouble();

                        separationLine();

System.out.println(toTable(classController.getAllClass_()));
                        System.out.print("Select class of the weapon by id: ");
                        temp = scanner.nextInt();
                        class_ = classController.getClass_(temp);
                        message = weaponController.createWeapon(name,
description, class_, dps);
                        System.out.println(message);
                        System.out.print("Input any letter to continue: ");
                        scanner.next();
                        break outer;
                    case 3:

System.out.println(toTable(weaponController.getAllWeapon()));
                        System.out.print("Input id of the weapon: ");
                        id = scanner.nextInt();

                        System.out.print("Input new name or press 0 to skip: ");
                        scanner.nextLine();
                        name = scanner.nextLine();
                        name = (name.charAt(0) == '0' && name.length() == 1) ?
null : name;

                        System.out.print("Input new description or press 0 to
skip: ");
                        description = scanner.nextLine();
                        description = (description.charAt(0) == '0' &&
description.length() == 1) ? null : description;

                        System.out.print("Input new dps or press 0 to skip: ");
                        dps = scanner.nextDouble();

                        separationLine();

System.out.println(toTable(classController.getAllClass_()));
                        System.out.print("Select class by id or press 0 to skip:
");

                        temp = scanner.nextInt();
                        class_ = (temp == 0) ? null :
classController.getClass_(temp);

                        if
(class_==null&&name==null&&description==null&&dps==0){
                            System.out.println("Weapon was not changed.");
                            System.out.print("Input any letter to continue: ");
                            scanner.next();
                            break outer;
                        }

                        message = weaponController.changeWeapon(id, name,
description, class_, dps);
                        System.out.println(message);
                        System.out.print("Input any letter to continue: ");
                        scanner.next();
                        break outer;
                    case 4:

System.out.println(toTable(weaponController.getAllWeapon()));
                        System.out.print("Input id of the weapon: ");
                        id = scanner.nextInt();
                        message = weaponController.deleteWeapon(id);
                        System.out.println(message);
                        System.out.print("Input any letter to continue: ");
                        scanner.next();
                        break outer;
                    case 9: break outer2;
                    case 0: System.exit(0);
                    default:
                        System.out.println("The input must be 1, 2, 3, 4, 9 or
0.");
                }
            }
        }
    }

                                        …

    public  void
separationLine(){System.out.print("\n\n________________________________________\
n");}

    public <T> String toTable(List<T> objects){
        String result = "\n";
        if (objects.get(0) instanceof Character){
            for (Object object : objects){
                Character character = (Character) object;
                result = result.concat("id: " + character.getId() + "\t name: "
+ character.getName() + "\n");
            }
        } else if (objects.get(0) instanceof Class_){
            for (Object object : objects){
                Class_ class_ = (Class_) object;
                result = result.concat("id: " + class_.getId() + "\t name: " +
class_.getName() + "\n");
            }
        } else if (objects.get(0) instanceof Ability) {
            for (Object object : objects){
                Ability class_ = (Ability) object;
                result = result.concat("id: " + class_.getId() + "\t name: " +
class_.getName() + "\n");
            }
        } else if (objects.get(0) instanceof Clothes) {
            for (Object object : objects){
                Clothes clothes = (Clothes) object;
                result = result.concat("id: " + clothes.getId() + "\t name: " +
clothes.getName() + "\n");
            }
        } else if (objects.get(0) instanceof Weapon){
            for (Object object : objects){
                Weapon weapon = (Weapon) object;
                result = result.concat("id: " + weapon.getId() + "\t name: " +
weapon.getName() + "\n");
            }
        }
        return result;
    }
    public <T extends Item> String toTable(List<T> items, Class_ class_){
        String result = "\n";
        for (Item item : items){
            if (item.getClass_().getId() == class_.getId()){
                result = result.concat("id: " + item.getId() + "\t name: " +
item.getName() + "\n");
            }
        }
        return result;
    }
}
```

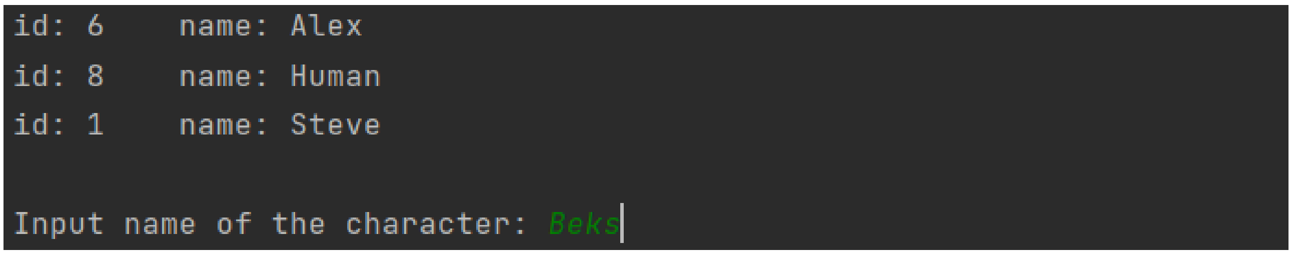
4. Program execution



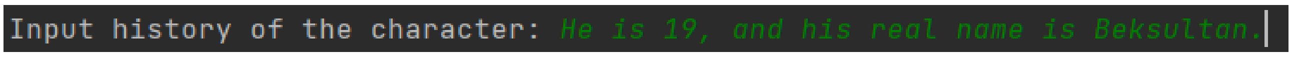
```
Select a table:
1: character
2: clothes
3: weapon
4: class
5: ability
---------------
0: exit

Input the number from 0 to 5: 1
```



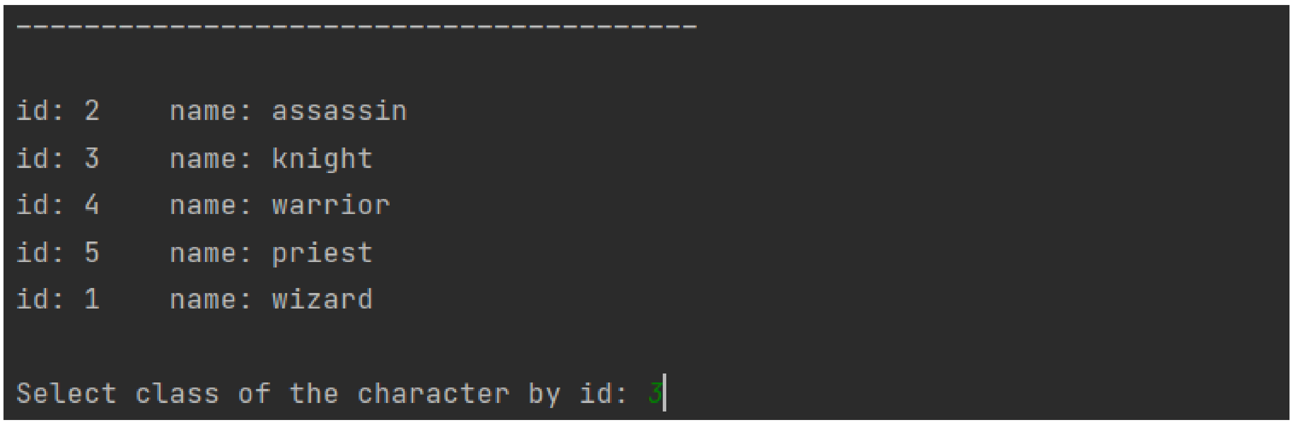
```
----------------------------------------
Select an action for table 'character':
1: get a record
2: create a record
3: change a record
4: delete a record
------------------
9: previous
0: exit

Input the number 1, 2, 3, 4, 9, or 0: 2
```



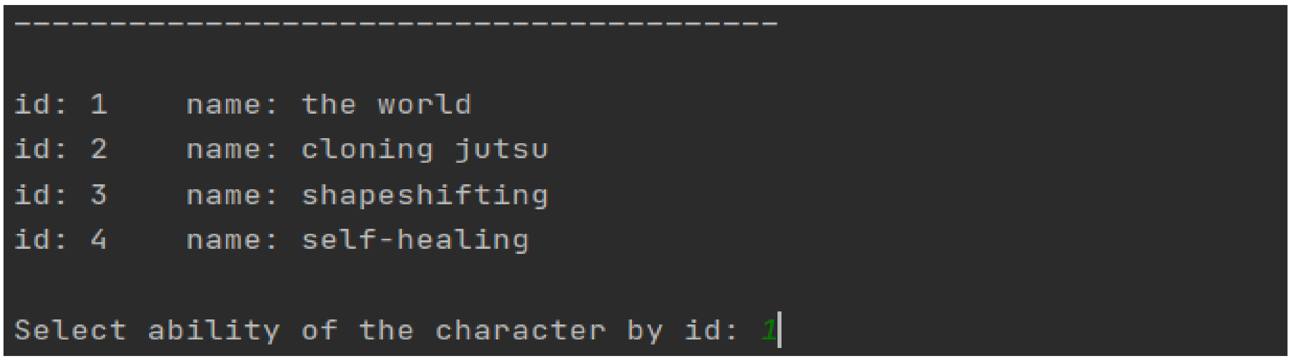
```
id: 6    name: Alex
id: 8    name: Human
id: 1    name: Steve

Input name of the character: Beks
```



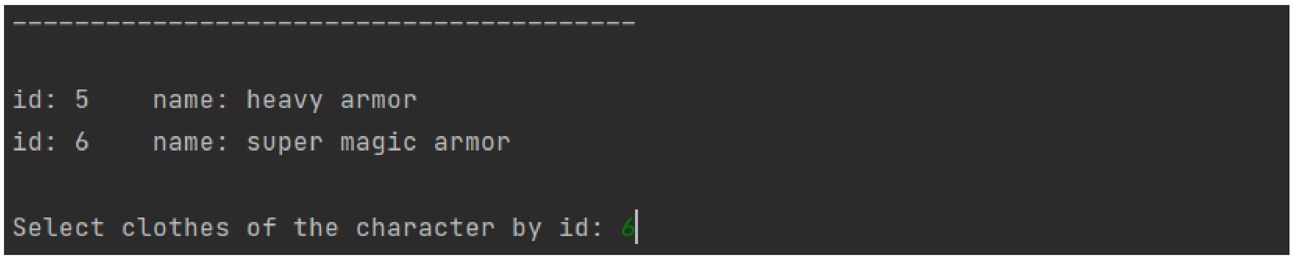
```
Input history of the character: He is 19, and his real name is Beksultan.
```



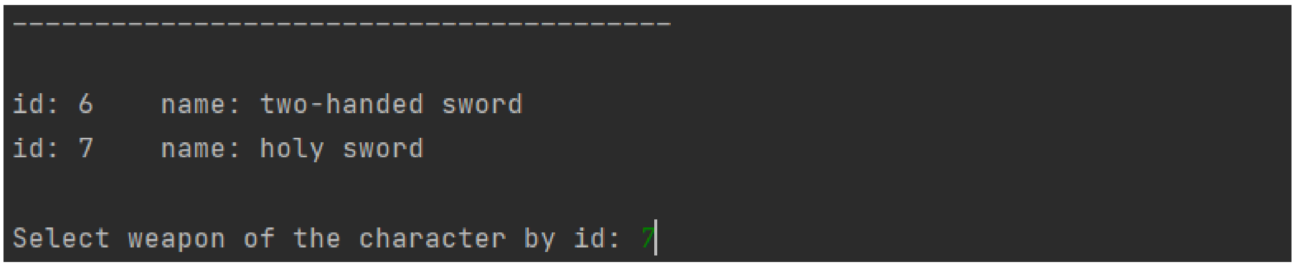
```
----------------------------------------


id: 2    name: assassin
id: 3    name: knight
id: 4    name: warrior
id: 5    name: priest
id: 1    name: wizard

Select class of the character by id: 3
```



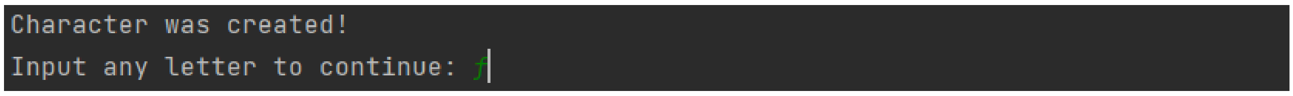
```
----------------------------------------

id: 1     name: the world
id: 2     name: cloning jutsu
id: 3     name: shapeshifting
id: 4     name: self-healing

Select ability of the character by id: 1
```



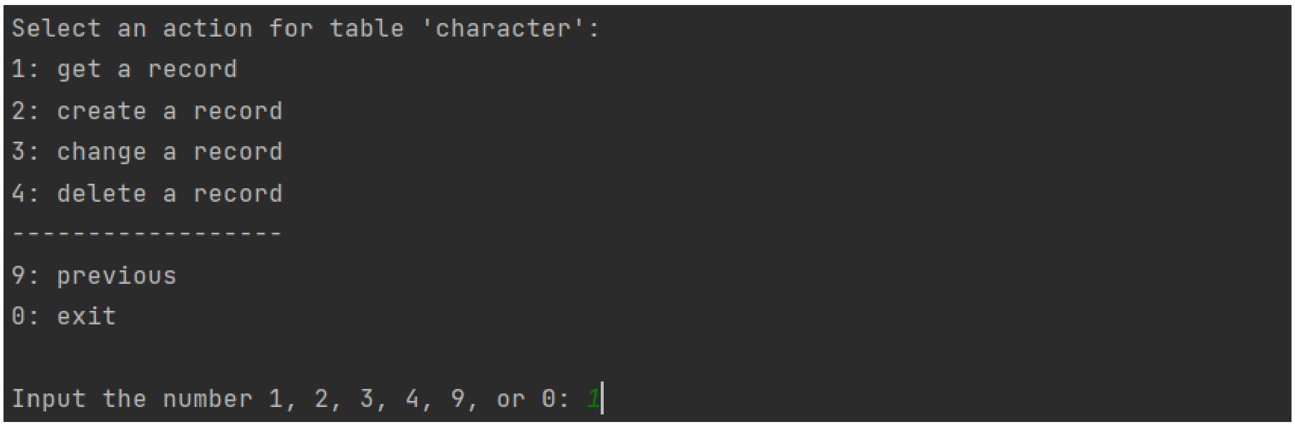
```
----------------------------------------

id: 5     name: heavy armor
id: 6     name: super magic armor

Select clothes of the character by id: 6
```



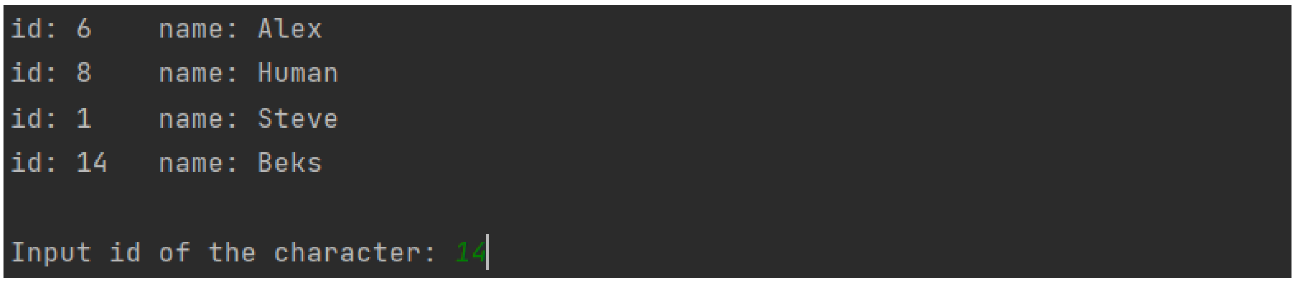
```
----------------------------------------

id: 6     name: two-handed sword
id: 7     name: holy sword

Select weapon of the character by id: 7
```



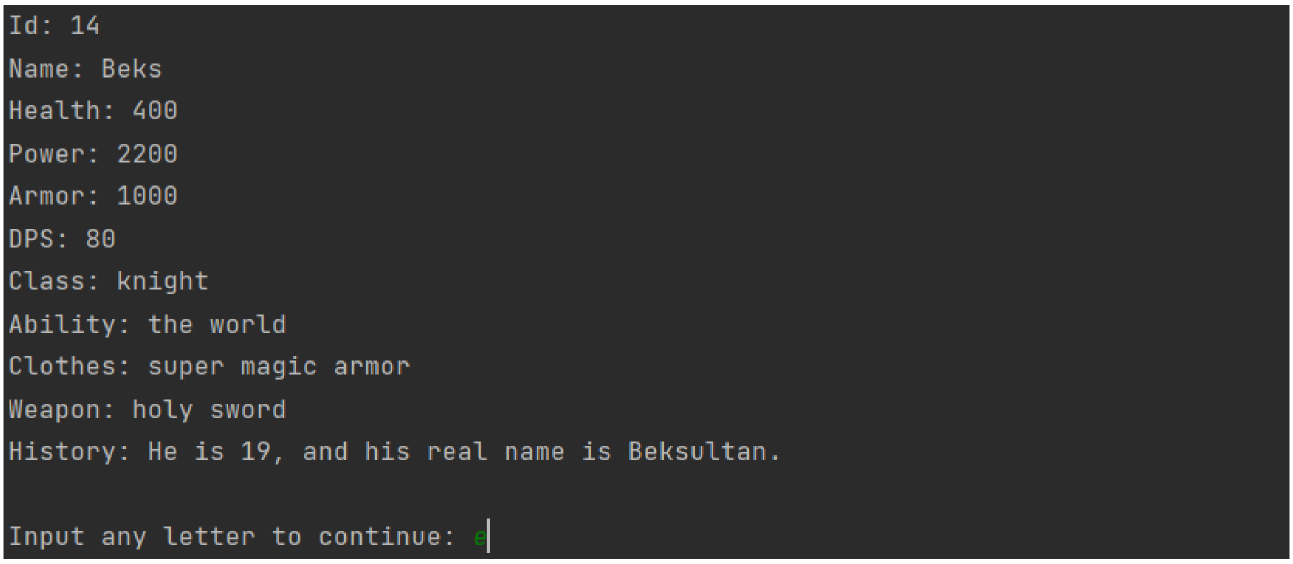
```
Character was created!
Input any letter to continue: f
```



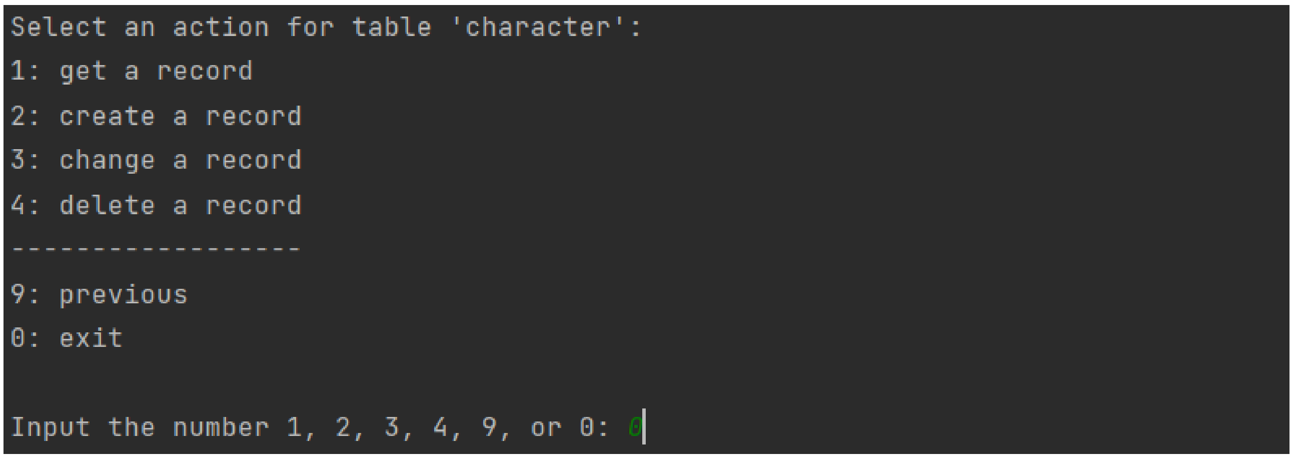
```
Select an action for table 'character':
1: get a record
2: create a record
3: change a record
4: delete a record
------------------
9: previous
0: exit

Input the number 1, 2, 3, 4, 9, or 0: 1
```



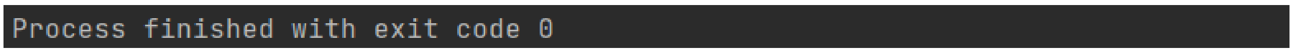
```
id: 6     name: Alex
id: 8     name: Human
id: 1     name: Steve
id: 14    name: Beks

Input id of the character: 14
```



```
Id: 14
Name: Beks
Health: 400
Power: 2200
Armor: 1000
DPS: 80
Class: knight
Ability: the world
Clothes: super magic armor
Weapon: holy sword
History: He is 19, and his real name is Beksultan.

Input any letter to continue: e
```



```
Select an action for table 'character':
1: get a record
2: create a record
3: change a record
4: delete a record
------------------
9: previous
0: exit

Input the number 1, 2, 3, 4, 9, or 0: 0
```



```
Process finished with exit code 0
```

5. Reflection from each team member

Nurlybai Beksultan:

My responsibilities were to find topic and idea, to write menu of the program and to write decent part of documentation. At the first, I thought that it would be easy to write the project, because it was not large and complex relatively. But when I was at the middle of the project, I find out that there are many features that I still need to write.

Kashshafidin Sherkhan:

My responsibilities included connecting the database to the project so that we can manage the database from within the program. I did that as it shown in the lecture. The first impression was somehow unpleasant when I started the video. But it wasn't all that complicated. Besides, it didn't require a very high intelligence.

Amanzhol Nurman:

My responsibilities included writing classes, abstract class and interfaces from the package 'entities'. Before starting programming, I had no idea about how to realize the project, but after forming the clear plan of hierarchy and relation of classes, coding became easier and more monotonous.

6. Conclusion

"Character creation mode" is pretty peculiar project. Many other projects were about employees, shops, products or others that are related to business and job. Meanwhile, our program was about video games that is not as serious as others, and theoretically have no benefits for humans. However, thanks to the freedom from thinking about the benefits for humans, our program has become more interesting and more complex than most other programs. It consists of five classes, two interfaces, one abstract class and one custom exception class. Moreover, we used such feature as method overloading, generics, casting and operator instanceof. All of these features are applied to make code easier to read, more organized and shorter. In general, this project was good simulation of group programming and working with OOP and gives good experience that will definitely help in the future when we will work with real commercial project.