Suyundykov Margulan

SE-2325

Software Design Patterns | Kapizov Dastan

Team Members: Margulan Suyundykov, Yernar Bukenbay

Link: https://github.com/IrentyM/CoolProject.git

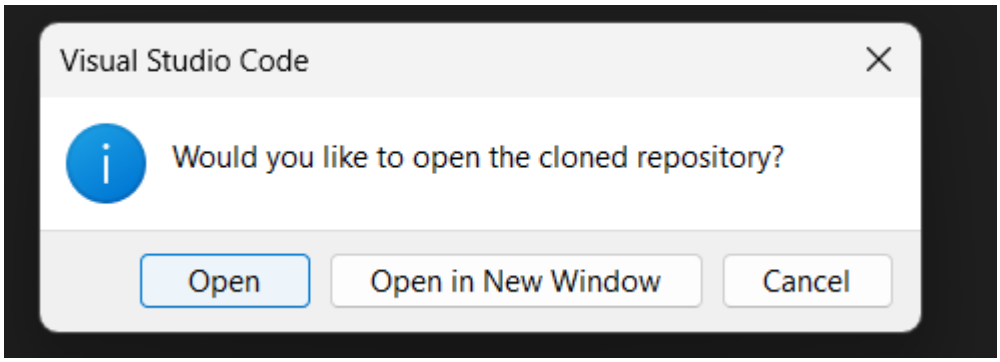
Kazakh Khanate: Struggle for Unity

Project information: This game is a turn-based grand strategy game set in the early 18th century, inspired by games like *Europa Universalis IV , Crusader Kings 3 and others*. The player assumes the role of a leader in one of the three Kazakh Juzes (tribal confederations) during the year 1721. The objective is to conquer, unite the Kazakh Khanate, and contend with neighboring powers. Players must navigate the complex geopolitical landscape, managing diplomacy, economy, and military campaigns to achieve their goals.

This game offers a deep and historically inspired strategic experience, allowing players to reshape the fate of the Kazakh Khanate and its role in the larger Central Asian region.

More information you can find in README

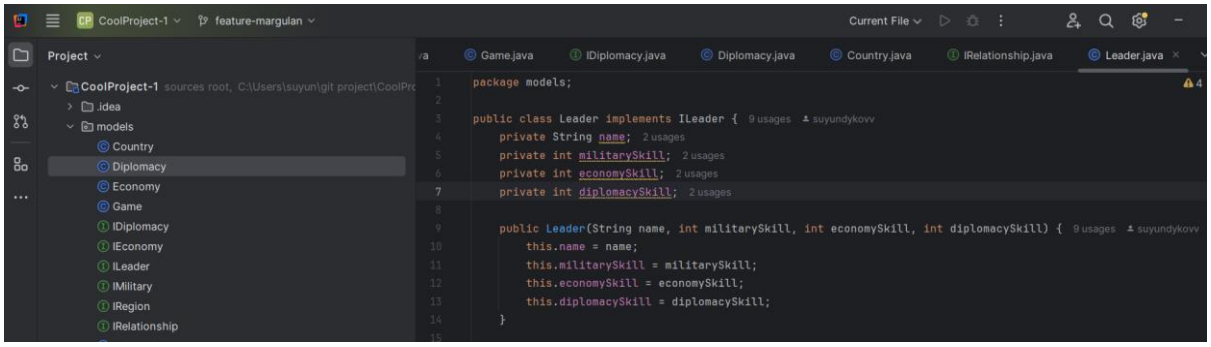
1.



Cloning repository in VC CODE

2. SOLID structure in project:

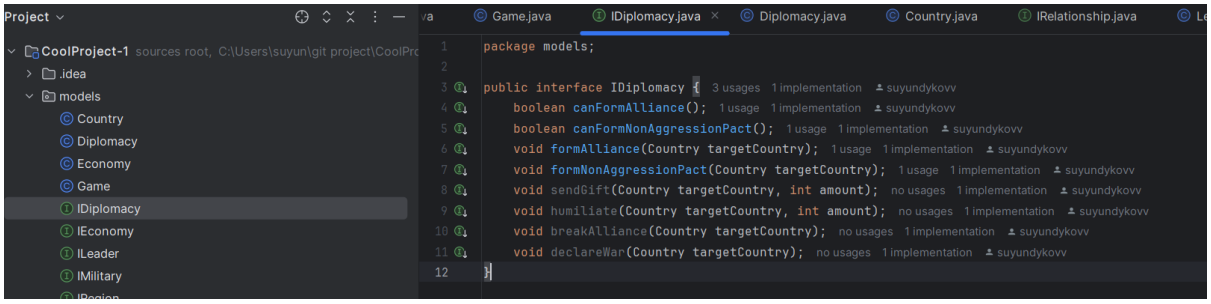
2.1



Separation of Concerns:

Each class and interface is responsible for a specific part of the game's functionality, adhering to the SRP.

2.2



Using of interfaces:

Interfaces are used for abstraction, allowing for easier testing and maintenance.

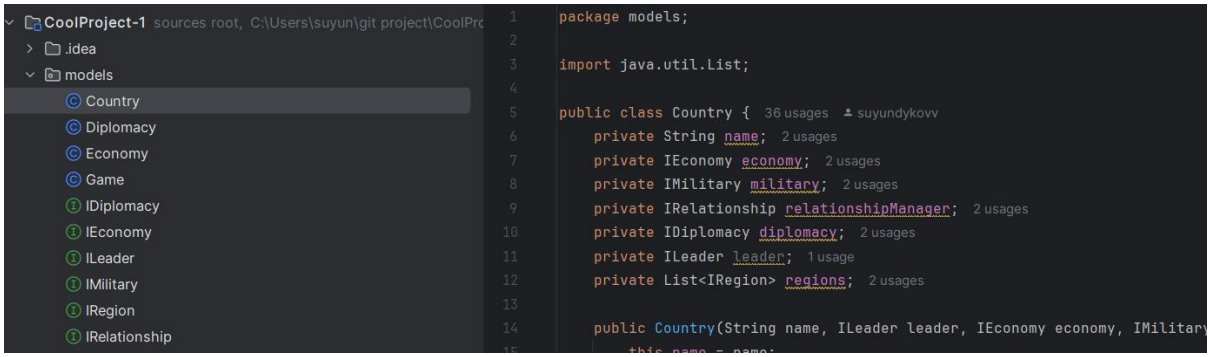
2.3

```java
package models;

import java.util.HashMap;
import java.util.Map;

public class RelationshipManager implements IRelationship {  1 usage  suyundykovv
    private Map<Country, RelationshipStatus> relationships = new HashMap<>();  2 usages

    @Override  8 usages  suyundykovv
    public void setRelationship(Country targetCountry, RelationshipStatus status) {
        relationships.put(targetCountry, status);
    }

    @Override  no usages  suyundykovv
    public RelationshipStatus getRelationship(Country targetCountry) {
        return relationships.getOrDefault(targetCountry, RelationshipStatus.NEUTRAL);
    }
}
```

Encapsulation of Relationships:

The RelationshipManager handles all relationship-related actions, allowing Country to focus on its own attributes and behaviors

2.4



Dependency Injection: Components such as IEconomy, IMilitary, and IRegion are injected into the Country class, promoting the DIP.

2.5

Clear Method Responsibilities: Methods are concise, focused on a single task, and provide meaningful outputs or logs.

3. Adding, Commiting, Pushing

```
PS C:\Users\suyun\git project\CoolProject-1>
PS C:\Users\suyun\git project\CoolProject-1> git add .
PS C:\Users\suyun\git project\CoolProject-1> git commit -m"rewrite whole project"
[feature-margulan df6bb84] rewrite whole project
 4 files changed, 211 insertions(+), 38 deletions(-)
PS C:\Users\suyun\git project\CoolProject-1> git add .
PS C:\Users\suyun\git project\CoolProject-1> git push
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 9.14 KiB | 1.02 MiB/s, done.
Total 11 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/IrentyM/CoolProject.git
   ca519c4..df6bb84  feature-margulan -> feature-margulan
PS C:\Users\suyun\git project\CoolProject-1>
```

Explaining code and logic of game:

1. Game Class

```java
package models;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Game {  3 usages  suyundykovv
    private List<Country> countries; // List of countries in the game  17 usages
    private Scanner scanner; // Scanner for user input  5 usages

    public Game() {  1 usage  suyundykovv
        this.countries = new ArrayList<>();
        this.scanner = new Scanner(System.in);
        initializeGame(); // Initialize game state
    }

    // Initialize game state and countries
    private void initializeGame() {  1 usage  suyundykovv
        // Initialize leaders

            // Initialize leaders
            ILeader peterI = new Leader( name: "Peter I",  militarySkill: 8,  economySkill: 7,  diplomacySkill: 8);
            ILeader yongzheng = new Leader( name: "Yongzheng Emperor",  militarySkill: 7,  economySkill: 8,  diplomacySkill: 7);
            ILeader tsewang = new Leader( name: "Tsewang Rabtan",  militarySkill: 8,  economySkill: 4,  diplomacySkill: 3);
            ILeader shahMohammed = new Leader( name: "Shah Mohammed",  militarySkill: 5,  economySkill: 6,  diplomacySkill: 7);
            ILeader kartAbulkhair = new Leader( name: "Kart-Abulkhair",  militarySkill: 4,  economySkill: 7,  diplomacySkill: 5);
            ILeader abulkhair = new Leader( name: "Abulkhair",  militarySkill: 8,  economySkill: 3,  diplomacySkill: 4);
            ILeader sherGaziKhan = new Leader( name: "Shergazi Khan",  militarySkill: 5,  economySkill: 5,  diplomacySkill: 6);
            ILeader muhammadRahim = new Leader( name: "Muhammad Rahim",  militarySkill: 4,  economySkill: 7,  diplomacySkill: 4);
            ILeader abdurahimBey = new Leader( name: "Abdurahim-bey",  militarySkill: 3,  economySkill: 6,  diplomacySkill: 6);

            // Initialize regions for each country
```

- Purpose: The main orchestration of the game. It handles user interactions, controls game flow, and manages the high-level operations of the game such as economy, military, and diplomacy for each country.
- Responsibilities:
    - Initializes the game setup (leaders, countries, regions, economies, militaries).
    - Provides a menu for the player to manage economy, military, and diplomacy.
    - Handles user input and applies game mechanics (like recruiting soldiers, forming alliances, calculating income, etc.).

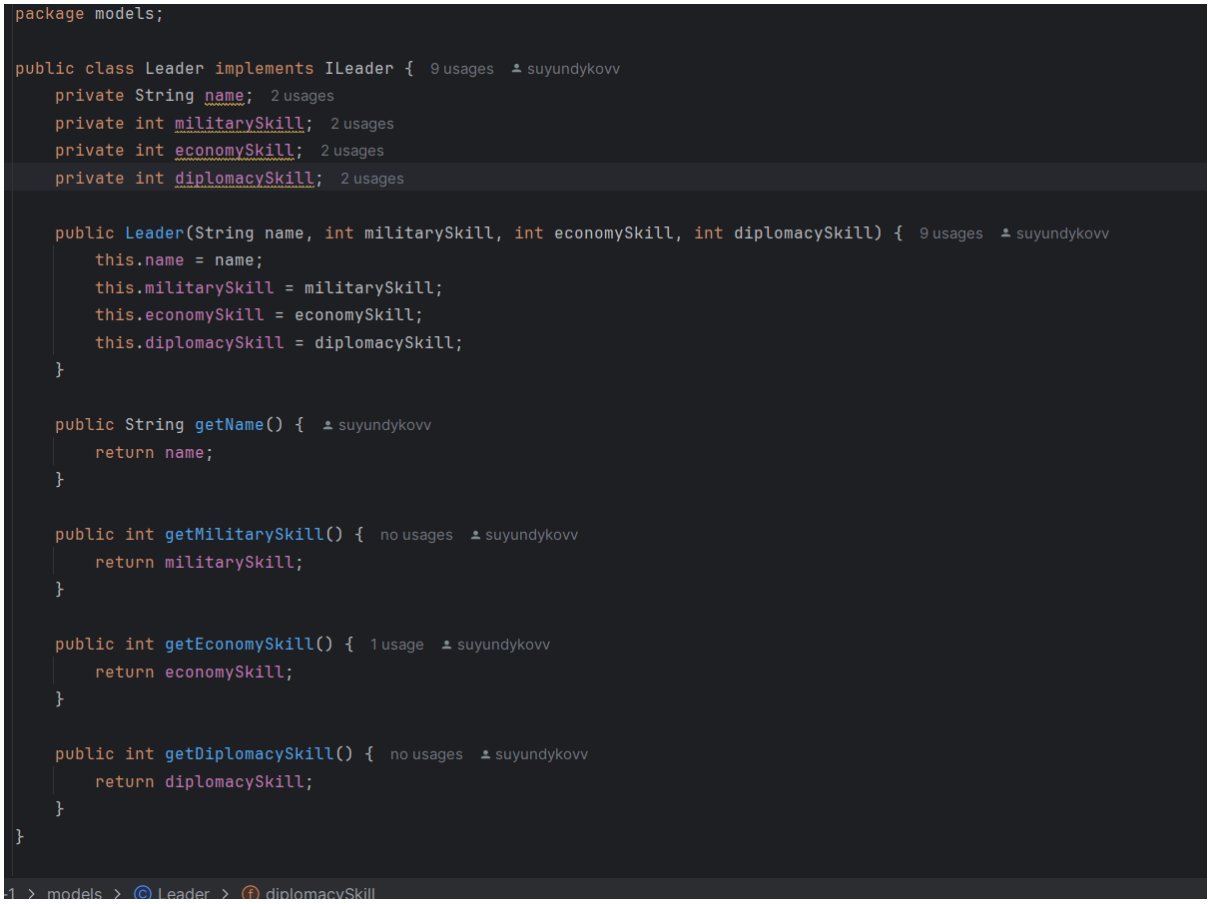
2. Country Class

```java
package models;

import java.util.List;

public class Country {  36 usages  suyundykovv
    private String name;  2 usages
    private IEconomy economy;  2 usages
    private IMilitary military;  2 usages
    private IRelationship relationshipManager;  2 usages
    private IDiplomacy diplomacy;  2 usages
    private ILeader leader;  1 usage
    private List<IRegion> regions;  2 usages

    public Country(String name, ILeader leader, IEconomy economy, IMilitary military, List<IRegion> regions) {  9 usages
        this.name = name;
        this.leader = leader;
        this.economy = economy;
        this.military = military;
        this.relationshipManager = new RelationshipManager();
        this.diplomacy = new Diplomacy( diplomacyPoints: 3, economy); // Example diplomacy points initialization
        this.regions = regions;
    }

    public String getName() {  suyundykovv
        return name;
    }

    public IEconomy getEconomy() {  2 usages  suyundykovv
        return economy;
    }

    public IMilitary getMilitary() {  2 usages  suyundykovv
```

- Purpose: Represents a country in the game.
- Responsibilities:
    - Holds references to a country's economy, military, diplomacy, and regions.

- Acts as a central entity for managing all operations related to the country (e.g., economy updates, diplomacy interactions, military recruitment).

3. Leader Class (Implements ILeader)



```
package models;

public class Leader implements ILeader {  9 usages  suyundykovv
    private String name;  2 usages
    private int militarySkill;  2 usages
    private int economySkill;  2 usages
    private int diplomacySkill;  2 usages

    public Leader(String name, int militarySkill, int economySkill, int diplomacySkill) {  9 usages  suyundykovv
        this.name = name;
        this.militarySkill = militarySkill;
        this.economySkill = economySkill;
        this.diplomacySkill = diplomacySkill;
    }

    public String getName() {  suyundykovv
        return name;
    }

    public int getMilitarySkill() {  no usages  suyundykovv
        return militarySkill;
    }

    public int getEconomySkill() {  1 usage  suyundykovv
        return economySkill;
    }

    public int getDiplomacySkill() {  no usages  suyundykovv
        return diplomacySkill;
    }
}
```

1 > models > Leader > diplomacySkill

- Purpose: Represents the leader of a country.
- Responsibilities:
    - Stores and provides access to the leader's skills (military, economic, and diplomatic).
    - Used by other components (e.g., economy or military) to influence their respective calculations (like economic growth or military recruitment).

4. Economy Class (Implements IEconomy)

```java
package models;

import java.util.List;

public class Economy implements IEconomy {  9 usages  suyundykovv
    private int money; // Current amount of money  10 usages
    private List<IRegion> regions; // List of regions to calculate region-based income  2 usages
    private ILeader leader; // Reference to leader for economic skill  2 usages

    private static final int DUCAT_VALUE = 10; // Value of each economic point in ducats  1 usage
    private static final int BASE_ECONOMIC_POINTS = 3; // Fixed base economic points  1 usage

    // Constructor
    public Economy(int initialMoney, List<IRegion> regions, ILeader leader) {  9 usages  suyundykovv
        this.money = initialMoney;
        this.regions = regions;
        this.leader = leader;
    }

    public void calculateIncome() {  1 usage  suyundykovv
        int income = calculateEconomicPoints() * DUCAT_VALUE; // Income calculation based on updated economic points
        money += income; // Add income to total money
        System.out.println("Income calculated: " + income + " ducats. Total money: " + money);
    }

    private int calculateEconomicPoints() {  1 usage  suyundykovv
        int totalEconomicPoints = BASE_ECONOMIC_POINTS + leader.getEconomySkill(); // Base + leader's skill
        for (IRegion region : regions) {
            totalEconomicPoints += region.getDevelopmentLevel(); // Add region development levels
        }
        return totalEconomicPoints;
    }
```

- Purpose: Manages the economic state of a country.
- Responsibilities:
    - Calculates and manages the country’s money and income based on economic points, regions, and the leader’s skill.
    - Provides methods to recruit soldiers (spending money) and deduct maintenance costs for soldiers.
    - Handles upgrades or changes in economic points when regions are improved.

5. Military Class (Implements IMilitary)

```java
package models;

public class Military implements IMilitary {  9 usages  suyundykovv
    private int soldiers; // Current number of soldiers  3 usages
    private int militaryPoints; // Military points for calculating recruits  3 usages
    private static final int RECRUITS_PER_MILITARY_POINT = 30; // Recruits per military point  1 usage

    // Constructor
    public Military(int initialSoldiers, int initialMilitaryPoints) {  9 usages  suyundykovv
        this.soldiers = initialSoldiers;
        this.militaryPoints = initialMilitaryPoints;
    }

    public void recruitSoldiers(int numberOfRecruits) {  1 usage  suyundykovv
        soldiers += numberOfRecruits; // Increase soldier count
    }

    public int getAvailableRecruits() {  no usages  suyundykovv
        return militaryPoints * RECRUITS_PER_MILITARY_POINT; // Total recruits available based on military points
    }

    public int getSoldiers() {  1 usage  suyundykovv
        return soldiers;
    }

    public void setMilitaryPoints(int militaryPoints) {  no usages  suyundykovv
        this.militaryPoints = militaryPoints;
    }
}
```

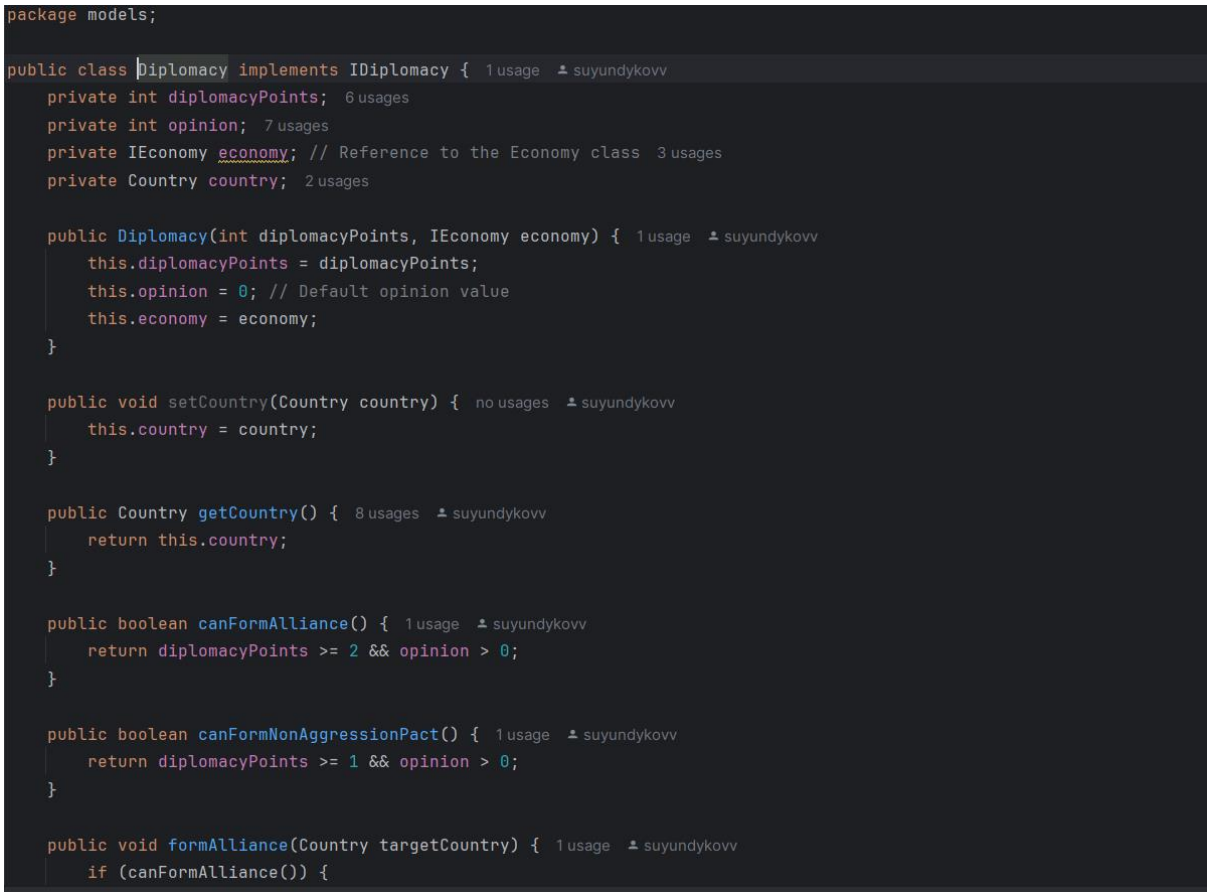
- Purpose: Manages the country's military strength.
- Responsibilities:
    - Tracks the number of soldiers and recruits available.
    - Provides methods to recruit soldiers and maintain the army.
    - Uses military points and the leader's military skill to calculate recruits and soldier strength.

6. Region Class (Implements IRegion)

```java
package models;

public class Region implements IRegion {  70 usages  suyundykovv
    private String name;  4 usages
    private int developmentLevel;  5 usages
    private String capital;  2 usages

    public Region(String name, int developmentLevel, String capital) {  70 usages  suyundykovv
        this.name = name;
        this.developmentLevel = developmentLevel;
        this.capital = capital;
    }

    public String getName() {  suyundykovv
        return name;
    }

    public int getDevelopmentLevel() {  1 usage  suyundykovv
        return developmentLevel;
    }

    public String getCapital() {  no usages  suyundykovv
        return capital;
    }

    public void upgradeDevelopmentLevel() {  no usages  suyundykovv
        if (developmentLevel < 10) {
            developmentLevel++;
            System.out.println(name + " upgraded to development level " + developmentLevel);
        } else {
            System.out.println(name + " is already at maximum development level.");
        }
```

- Purpose: Represents a region or territory within a country.
- Responsibilities:
    - Holds the name and development level of a region, which impacts the country's economic output and the number of soldiers it can support.
    - Development level is key in determining the economic and military contributions of the region to the country.

7. Diplomacy Class (Implements IDiplomacy)



```
package models;

public class Diplomacy implements IDiplomacy {  1 usage  suyundykovv
    private int diplomacyPoints;  6 usages
    private int opinion;  7 usages
    private IEconomy economy; // Reference to the Economy class  3 usages
    private Country country;  2 usages

    public Diplomacy(int diplomacyPoints, IEconomy economy) {  1 usage  suyundykovv
        this.diplomacyPoints = diplomacyPoints;
        this.opinion = 0; // Default opinion value
        this.economy = economy;
    }

    public void setCountry(Country country) {  no usages  suyundykovv
        this.country = country;
    }

    public Country getCountry() {  8 usages  suyundykovv
        return this.country;
    }

    public boolean canFormAlliance() {  1 usage  suyundykovv
        return diplomacyPoints >= 2 && opinion > 0;
    }

    public boolean canFormNonAggressionPact() {  1 usage  suyundykovv
        return diplomacyPoints >= 1 && opinion > 0;
    }

    public void formAlliance(Country targetCountry) {  1 usage  suyundykovv
        if (canFormAlliance()) {
```

- Purpose: Manages diplomatic relationships between countries.
- Responsibilities:
    - Provides methods to form alliances or non-aggression pacts between countries.
    - Tracks the current relationship status (e.g., neutral, allied, at war) with other countries.
    - Manages the opinion values between countries, which can affect diplomatic actions.

8. RelationshipStatus Enum

```
package models; // or use enums if you have that package

public enum RelationshipStatus {  14 usages  suyundykovv
    NEUTRAL,      // Default state, no active diplomacy  3 usages
    ALLIED,       // Allied with the country  2 usages
    AT_WAR,       // At war with the country  2 usages
    PACT          // Non-aggression pact  2 usages
}
```

- Purpose: Defines the various possible states of relationships between countries.
- Responsibilities:
  - Provides predefined constants for diplomatic status: Neutral, Allied, At War, and Non-Aggression Pact.
  - Used by the Diplomacy class to manage relationship statuses between countries.

9. ILeader, IEconomy, IMilitary, IRegion, and IDiplomacy Interfaces

```
package models;

public interface ILeader {  14 usages  1 implementation  suyundykovv
    String getName();  1 implementation  suyundykovv
    int getMilitarySkill();  no usages  1 implementation  suyundykovv
    int getEconomySkill();  1 usage  1 implementation  suyundykovv
    int getDiplomacySkill();  no usages  1 implementation  suyundykovv
}
```

- Purpose: Provide abstractions for different domain areas like leader, economy, military, region, and diplomacy.
- Responsibilities:
  - These interfaces define the core operations for each respective component.
  - Promote the dependency inversion principle by ensuring that high-level modules (Game and Country) do not depend on specific implementations but on abstractions.

Terminal Output:

1.Main

```java
import models.Game;

public class Main {  suyundykovv
    public static void main(String[] args) {  suyundykovv
        Game game = new Game();
        game.start();
    }
}
```

2. Choose country

```
0. Russian Empire
1. Qing Dynasty
2. Zhungar Khanate
3. Middle Juz
4. Uly Juz
5. Kishi Juz
6. Xiva
7. Bukhara
8. Kokand
```

3. Menu

```
1
=== Game Menu ===
1. Manage Economy
2. Manage Military
3. Manage Diplomacy
0. Exit
Enter your choice:
```

4. Choosing country for diplomatic interactions:

```
Enter your choice: 3
Manage Diplomacy - Select a Country:
0. Russian Empire
1. Qing Dynasty
2. Zhungar Khanate
3. Middle Juz
4. Uly Juz
5. Kishi Juz
6. Xiva
7. Bukhara
8. Kokand
```

5. Diplomatic Interactions

```
5
1. Form Alliance
2. Form Non-Aggression Pact
0. Back to Menu
```