Table Of Content

Package axisandallies

Class Summary

Faction

MainGame

Territory

TerritoryTest

Unit

---

axisandallies

Class Faction

```
java.lang.Object
    |
    +--axisandallies.Faction
```

< Constructors > < Methods >

---

public class **Faction**
extends java.lang.Object

Constructors

Faction

public **Faction**()

Faction

```
public  Faction(int faction,
               int playerID,
               int bank,
               int income)
```

Constructor for the faction class

Parameters:

faction - which faction is this instance of the class
playerID - the playerID playing this class
bank - the number of IPCs the player has
income - the current income of the player

Methods

decreaseIncome

```
public boolean decreaseIncome(int newIncome)
```

---

decreasePlayerBank

```
public boolean decreasePlayerBank(int amount)
```

---

getFaction

```
public int getFaction()
```

---

getIncome

```
public int getIncome()
```

---

getPlayerBank

```
public int getPlayerBank()
```

---

getPlayerID

```
public int getPlayerID()
```

---

getResearch

```
public boolean[] getResearch()
```

---

increaseIncome

```
public boolean increaseIncome(int newIncome)
```

---

increasePlayerBank

```
public boolean increasePlayerBank(int amount)
```

---

rollResearch

```
public boolean rollResearch(int amtWager,
                            int researchAttempted)
```

Rolling for research for a specific faction. Research is performed by rolling dice for the indicated number of the research (i.e. Fighter jets succeeds on a roll of a 1). Dice is purchased in increments of 5 IPC each.

Parameters:

amtWager - the amount of IPC spent on dice
researchAttempted - the index number of the research being attempted

Returns:

boolean to indicate whether the method completed correctly

---

setFaction

```
public boolean setFaction(int newFaction)
```

---

setPlayerID

```
public boolean setPlayerID(int newPlayerID)
```

---

setResearch

```
public boolean setResearch(int researchPos)
```

---

axisandallies

Class MainGame

```
java.lang.Object
    |
    +--axisandallies.MainGame
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

public class **MainGame**
extends java.lang.Object

Fields

DEFAULT_MIN_VALUE

```
static final int DEFAULT_MIN_VALUE
```

MAX_PLAYERS

```
static final int MAX_PLAYERS
```

MAX_RESEARCH

```
static final int MAX_RESEARCH
```

MAX_TERRITORIES

```
static final int MAX_TERRITORIES
```

NUMBER_OF_PLAYERS

static int **NUMBER_OF_PLAYERS**

WATER_TERRITORY_COUNT

static final int **WATER_TERRITORY_COUNT**

factionList

static axisandallies.Faction[] **factionList**

territoryList

static axisandallies.Territory[] **territoryList**

Constructors

MainGame

public **MainGame**()

Methods

assignPlayerToFaction

public static void **assignPlayerToFaction**(int numPlayers,
                                          int userChoice,
                                          int playerNumber)

Assign the player to a faction

Parameters:

numPlayers - the number of players in the game.
userChoice - the team the user has selected to play.
playerNumber - the player number of the current user

developWeapons

```
public static void developWeapons(int currentPlayer)
```

> Phase 1: Select research to develop this turn Require user to select option to research (or none). Require user to enter the number of dice to purchase. Roll for research, then complete the phase.
>
> **Parameters:**
>
> > currentPlayer - the current player whose turn it is.

---

initializePlayers

```
public static void initializePlayers()
```

> Initialize the factions with starting values. Initialize number of players in the game. Assign players to teams.

---

initializeTerritories

```
public static void initializeTerritories()
```

> Initialize all territories to their default values and owners Initialization according to 2004 Revised Edition Ruleset

---

main

```
public static void main(java.lang.String[] args)
```

---

playerSelectMenu

```
public static void playerSelectMenu(int numberOfPlayers)
```

> Menu output to display teams based on the number of players.
>
> **Parameters:**
>
> > numberOfPlayers - the number of players in the game.

---

printTerrs

```
public static void printTerrs()
```

> Helper function to print all territory names

---

researchMenu

```
public static void researchMenu(int menuPage,
                                int currentPlayer)
```

> Output menu options for the research phase
>
> **Parameters:**
>
>> menuPage - the page to be displayed.
>> currentPlayer - the current player whose turn it is.

---

axisandallies

Class Territory

```
java.lang.Object
   |
   +--axisandallies.Territory
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

---

public class **Territory**
extends java.lang.Object

Fields

MAX_FACTION

```
public static final int MAX_FACTION
```

MIN_VALUE

```
public static final int MIN_VALUE
```

Constructors

Territory

```
public  Territory()
```

Territory

```
public  Territory(java.lang.String name,
                  int faction,
                  int value,
                  boolean landTerritory,
                  boolean isNeutral,
                  boolean isCapital,
                  boolean isVictory,
                  int infantry,
                  int artillery,
                  int tank,
                  int fighter,
                  int bomber,
                  int battleship,
                  int aircraftCarrier,
                  int transport,
                  int submarine,
                  int destroyer,
                  boolean antiaircraftGun,
                  boolean industrialComplex)
```

Territory constructor. Format: Name, faction, territory value, land or sea, neutral or playable, has a capital city, has a victory city, number of the following units: infantry, artillery, tank, fighter, bomber, battleship, aircraft carrier, transport, submarine, destroyer, has an anti aircraft gun, has an industrial complex

Methods

getAircraftCarrier

```
public int[] getAircraftCarrier()
```

---

getAntiAircraft

```
public boolean getAntiAircraft()
```

---

getArtillery

```
public int[] getArtillery()
```

---

getBattleship

```
public int[] getBattleship()
```

getBomber

public int[] **getBomber**()

getCapital

public boolean **getCapital**()

getConnections

public java.lang.String[] **getConnections**()

getConnectionsNumber

public int **getConnectionsNumber**()

getDestroyer

public int[] **getDestroyer**()

getFaction

public int **getFaction**()

getFight

public int[] **getFight**()

getIndustrialComplex

```
public boolean getIndustrialComplex()
```

---

getInfantry

```
public int[] getInfantry()
```

---

getIsConnected

```
public boolean getIsConnected(java.lang.String territory)
```

---

getIsLand

```
public boolean getIsLand()
```

---

getIsNeutral

```
public boolean getIsNeutral()
```

---

getName

```
public java.lang.String getName()
```

---

getSubmarine

```
public int[] getSubmarine()
```

---

getTank

```
public int[] getTank()
```

getTransport

```
public int[] getTransport()
```

getValue

```
public int getValue()
```

getVictory

```
public boolean getVictory()
```

setAircraftCarrier

```
public boolean setAircraftCarrier(int factionPos,
                                  int count)
```

setAntiAircraft

```
public boolean setAntiAircraft(boolean AA)
```

setArtillery

```
public boolean setArtillery(int factionPos,
                            int count)
```

setBattleship

```
public boolean setBattleship(int factionPos,
                             int count)
```

setBomber

```
public boolean setBomber(int factionPos,
                         int count)
```

---

setCapital

```
public boolean setCapital(boolean cap)
```

---

setDestroyer

```
public boolean setDestroyer(int factionPos,
                            int count)
```

---

setFaction

```
public boolean setFaction(int newFaction)
```

---

setFighter

```
public boolean setFighter(int factionPos,
                          int count)
```

---

setIndustrialComplex

```
public boolean setIndustrialComplex(boolean IC)
```

---

setInfantry

```
public boolean setInfantry(int factionPos,
                           int count)
```

---

setIsLand

```
public boolean setIsLand(boolean isLand)
```

setIsNeutral

```
public boolean setIsNeutral(boolean isNeutral)
```

setLandUnitParamterTest

```
public boolean setLandUnitParamterTest(int factionPos,
                                       int count)
```

Helper function to test valid parameters for updating the number of land units in a territory

Parameters:

factionPos - the faction that owns the units to be updated

count - the number of units remaining in the territory

Returns:

boolean whether the update was successful or not

setName

```
public boolean setName(java.lang.String name)
```

setNewConnection

```
public boolean setNewConnection(java.lang.String territory)
```

List of the number of connecting territories to the current territory

Parameters:

territory - the name of the territory to set connections

Returns:

boolean to indicate that the connections were set correctly

setSeaUnitParameterTest

```
public boolean setSeaUnitParameterTest(int factionPos,
                                       int count)
```

> Helper function to test valid parameters for updating the number of sea units in a territory
>
> **Parameters:**
>> factionPos - the faction that owns the units to be updated
>> count - the number of units remaining in the territory
>
> **Returns:**
>> boolean whether the update was successful or not

---

setSubmarine

```
public boolean setSubmarine(int factionPos,
                            int count)
```

---

setTank

```
public boolean setTank(int factionPos,
                       int count)
```

---

setTransport

```
public boolean setTransport(int factionPos,
                            int count)
```

---

setValue

```
public boolean setValue(int newVal)
```

---

setVictory

```
public boolean setVictory(boolean vic)
```

---

updateLandUnits

```
public boolean updateLandUnits(int factionPos,
                               int infantryCount,
                               int artilleryCount,
                               int tankCount,
                               int fighterCount,
                               int bomberCount)
```

Update land units of all types in a single territory

Parameters:

factionPos - the faction owner of the units to be updated
infantryCount - the number of infantry remaining
artilleryCount - the number of artillery remaining
tankCount - the number of tanks remaining
fighterCount - the number of fighters remaining
bomberCount - the number of bombers remaining

Returns:

boolean to indicate whether update was successful or not

---

updateSeaUnits

```
public boolean updateSeaUnits(int factionPos,
                              int battleshipCount,
                              int aircraftCarrierCount,
                              int transportCount,
                              int submarineCount,
                              int destroyerCount)
```

Update sea units of all types in a single territory

Parameters:

factionPos - the faction owner of the units to be updated
battleshipCount - the number of battleships remaining
aircraftCarrierCount - the number of aircraft carriers remaining
transportCount - the number of transports remaining
submarineCount - the number of submarines remaining
destroyerCount - the number of destroyers remaining

Returns:

boolean to indicate whether update was successful or not

---

axisandallies

Class TerritoryTest

```
java.lang.Object
    |
    +--axisandallies.TerritoryTest
```

< [Constructors](#) > < [Methods](#) >

---

public class **TerritoryTest**
extends java.lang.Object

Constructors

TerritoryTest

public **TerritoryTest**()

Methods

testFaction

public void **testFaction**()

---

testIC

public void **testIC**()

---

testInfantry

public void **testInfantry**()

---

testName

public void **testName**()

Class Unit

```
java.lang.Object
     |
     +--axisandallies.Unit
```

< [Constructors](#) > < [Methods](#) >

public class **Unit**
extends java.lang.Object

Constructors

Unit

public **Unit**()

Methods

getAttack

public int **getAttack**()

getCost

public int **getCost**()

getDefense

public int **getDefense**()

getFaction

public int **getFaction**()

getMove

```
public int getMove()
```

getType

```
public int getType()
```

setAttack

```
public boolean setAttack(int newAttack)
```

setCost

```
public boolean setCost(int newCost)
```

setDefense

```
public boolean setDefense(int newDefense)
```

setFaction

```
public boolean setFaction(int newFaction)
```

setMove

```
public boolean setMove(int newMove)
```

setType

```
public boolean setType(int newType)
```

INDEX

T

U

W