Table Of Content

Package axisandallies

Class Summary

Attack

Faction

MainGame

Territory

TerritoryTest

Unit

---

axisandallies

Class Attack

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

< Constructors > < Methods >

---

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

Author:
FlintTD

Constructors

Attack

```
public  Attack()
```

Attack

```
public  Attack(Territory attacker,
               Territory defender,
               int aInfantry,
               int dInfantry,
               int aArtillery,
               int dArtillery,
               int aTank,
               int dTank,
               int aFighter,
               int dFighter,
               int aBomber,
               int dBomber,
               int aBattleship,
               int dBattleship,
               int aAircraftCarrier,
               int dAircraftCarrier,
               int aTransport,
               int dTransport,
               int aSubmarine,
               int dSubmarine,
               int aDestroyer,
               int dDestroyer,
               boolean antiaircraftGun)
```

Constructor for the Attack class. Format: attacking territory, defending territory, number of the following units: attacking infantry, defending infantry, attacking artillery, defending artillery, attacking tanks, defending tanks, attacking fighters, defending fighters, attacking bombers, defending fighters, attacking battleships, defending battleships, attacking carriers, defending carriers, attacking transports, defending transports, attacking submarines, defending submarines, attacking destroyers, defending destroyers.

Methods

getAircraftCarrier

```
public int getAircraftCarrier(java.lang.String type)
```

---

getArtillery

```
public int getArtillery(java.lang.String type)
```

---

getBattleship

```
public int getBattleship(java.lang.String type)
```

---

getBomber

public int **getBomber**(java.lang.String type)

---

getDestroyer

public int **getDestroyer**(java.lang.String type)

---

getFighter

public int **getFighter**(java.lang.String type)

---

getInfantry

public int **getInfantry**(java.lang.String type)

---

getSubmarine

public int **getSubmarine**(java.lang.String type)

---

getTank

public int **getTank**(java.lang.String type)

---

getTransport

public int **getTransport**(java.lang.String type)

---

setAircraftCarrier

```
public void setAircraftCarrier(int attackers,
                               int defenders)
```

---

setArtillery

```
public void setArtillery(int attackers,
                         int defenders)
```

---

setAttacker

```
public void setAttacker(Territory wokka)
```

---

setBattleship

```
public void setBattleship(int attackers,
                          int defenders)
```

---

setBomber

```
public void setBomber(int attackers,
                      int defenders)
```

---

setDefender

```
public void setDefender(Territory flokka)
```

---

setDestroyer

```
public void setDestroyer(int attackers,
                         int defenders)
```

---

setFighter

```
public void setFighter(int attackers,
                       int defenders)
```

---

setInfantry

```
public void setInfantry(int attackers,
                        int defenders)
```

---

setSubmarine

```
public void setSubmarine(int attackers,
                         int defenders)
```

---

setTank

```
public void setTank(int attackers,
                    int defenders)
```

---

setTransport

```
public void setTransport(int attackers,
                         int defenders)
```

---

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

allyFaction

public static boolean **allyFaction**(int currentFaction,
int otherFaction)

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

---

collectIncome

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

Phase 5: Collect income. Update the the current player's bank with the current player's income. Income should be updated in combat phase when the territory is won or lost.

Parameters:

currentPlayer - is the current player whose turn it is.

---

combatMove

```
public static void combatMove(int currentFaction,
                              Unit units,
                              int numberOfUnits,
                              Territory attackingTerritory,
                              Territory defendingTerritory)
```

A helper function to facilitate a single combat move for any single-type group of units Note that many units have more than one combat move per turn, this is not accounted for here

Parameters:

currentFaction - is the faction moving their units
units - are the unit type moving
numberOfUnits - is the number of units moving //@param unitMoves is the number of territories a unit can move (fungible)
attackingTerritory - is the territory the units are coming from
defendingTerritory - is the territory the units are going to

---

combatMoveAndCombat

```
public static void combatMoveAndCombat(int currentPlayer,
                                        int currentFaction)
```

Phase 3: Move units from friendly territories into hostile territories Require user to select territories and declare units to invade them. Require user to enter the friendly and enemy territory names. Move units into territories and end the phase.

Parameters:

currentPlayer - the current player whose turn it is.
currentFaction - is the faction of the player.

---

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.

---

flacked

```
public static int flacked(Territory territory,
                          int faction)
```

Helper function to resolve anti-aircraft opening fire

Parameters:

territory - is the territory that does the flacking
faction - is the faction being flacked Citation:
http://introcs.cs.princeton.edu/java/13flow/RollDie.java.html

---

getTerritoryFromName

```
public static Territory getTerritoryFromName(java.lang.String nameString)
```

---

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.

---

resolveCombat

`public static void` **`resolveCombat`**`(Territory territory)`

Helper function to resolve combats for units in hostile territories

Parameters:

territory - is the territory that needs resolution

---

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

getAircraftCarriers

```
public int getAircraftCarriers(int faction)
```

---

getAntiAircraft

```
public boolean getAntiAircraft()
```

---

getArtillery

```
public int getArtillery(int faction)
```

---

getBattleships

```
public int getBattleships(int faction)
```

getBombers

public int **getBombers**(int faction)

getCapital

public boolean **getCapital**()

getConnections

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

getConnectionsNumber

public int **getConnectionsNumber**()

getDestroyers

public int **getDestroyers**(int faction)

getFaction

public int **getFaction**()

getFighters

public int **getFighters**(int faction)

getIndustrialComplex

public boolean **getIndustrialComplex**()

---

getInfantry

public int **getInfantry**(int faction)

---

getIsConnected

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

---

getIsFactionUnits

public boolean **getIsFactionUnits**(int faction)

---

getIsLand

public boolean **getIsLand**()

---

getIsNeutral

public boolean **getIsNeutral**()

---

getName

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

---

getSubmarines

public int **getSubmarines**(int faction)

getTanks

```
public int getTanks(int faction)
```

getTransports

```
public int getTransports(int faction)
```

getValue

```
public int getValue()
```

getVictory

```
public boolean getVictory()
```

setAircraftCarriers

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

setAntiAircraft

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

setArtillery

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

setBattleships

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

---

setBombers

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

---

setCapital

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

---

setDestroyers

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

---

setFaction

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

---

setFighters

```
public boolean setFighters(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

---

setSubmarines

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

---

setTanks

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

---

setTransports

```
public boolean setTransports(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()
```

---

axisandallies

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 java.lang.String **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**(java.lang.String newType)

INDEX

M

N

P

R

S

T

U

W