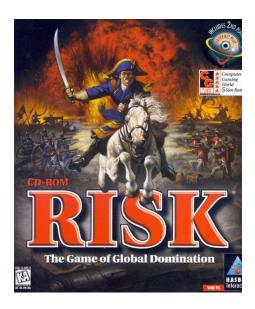
עבודה ב OOP 31631 סמסטר א' התשע"ט, עבודה מסכמת

RISK-risk the game of global משחק- סיכון domination



תאריך ההגשה: 25/02/2019

שמות הסטודנטים:

orisade12@gmail.com 318262128 ..ז. אורי שדה, ת.ז.

gal287199@gmail.com 206232506 גל אורן, ת.ז.

בהנחיית המרצה: יעל ארז

| | ציון: |
|------|-------|
| | |

<u>תאור הספריה:</u>

הספרייה risk מנהלת משחק risk ומספקת מחלקות ומתודות המתאימות למשחק.

- : הספרייה מאפשרת
- אפשרות לבחור את מספר השחקנים בין 2-4.
- אפשרות לקבוע את סדר התורות ע"י הטלת קובייה.
 - יצירה של מפה של 49 ארצות ו6 יבשות
- חלוקה התחלתית של חיילים לשחקנים על מנת שיאכלסו את המדינות- משתנה לפי מספר השחקנים.
- ניהול סבב כיבושים ראשוני ע"י השחקנים עד שהמפה מתמלאת וסבב נוסף לחיזוק השליטה ע"י השחקנים.
 - . draft,attack,reinforcment ניהול שלושת השלבים שמתבצעים במהלך המשחק
 - חלוקת קלפי כיבושים לשחקנים ומתן חיילי בונוס בהתאם לשחקנים.
 - מאפשרת שמירה בכל שלב ושחזור המשחק לכל נקודה.
- הצגת סטטיסטיקה על המשחקים-השחקן המצטיין השחקן עם הכי הרבה הפסדים –
 השחקן ששיחק הכי הרבה זמן ,השחקן ששיחק הכי הרבה משחקים ,זמן ממוצע של
 משחק בשניות
- * הספרייה מלווה בליווי של ממשק גרפי ידודתי המציג את הנתונים שמנוהלים על ידי הספריה כמו כן ישנו שיר רקע המלווה את השיר וצלילם לפפעולות שונות במשחק(כמו כיבוש ארץ).
 - Encapsulation, Inheritance, Templates, שימוש בנושאים אשר נלמדו בקורס •
 Operator overloading, STL

המחלקות שמומשו במסגרת הספרייה:

מחלקה אבסטרקטית לשלב משחק-Stage

–Place map מחלקה לשלב המשחק הראשוני-מיקום החיילים עד שהמפה מלאה

Place troops מחלקה שאחראית על ניהול השלב השני באתחול המפה- עד תום מיקום החיילים על פני המפה

. מחלקה זו אחראית על השלב הראשון בסבב המשחק—Draft

Attack מחלקה זו אחראית על שלב התקיפה בסבב המשחק

-Reinforcement מחלקה זו אחראית על השלב השלישי בסבב משחק של שחקן.

הראשי guia המחלקה הראשית בה מתנהל המשחק ו-

בחלקה הממשת שחקן- שומרת נתונים עבורו-Player

במחלקה המממשת קלף משחק – Card

בשת בשת הממשת יבשת Land

-Country מחלקה המממשת מדינה.

של תפריט המשחק gui מחלקה המנהלת את -Menu

. templat – מחלקה המנהלת מבנה נתונים של וקטור

class templet **-Playerstas** המנהלת את מבנה הנתונים של הסטטיסטיקות של השחקנים . **-Playerstatwrap** מחלקה ששומרת את המשתנים לגבי סטטיסטיקות של שחקן. **-Statsguimain** של הסטטיסטיקה.

: cube המחלקה

המחלקה מנהלת אובייקט של קוביה –מגרילה מספרים בין 1 ל6 - עבור השלב הראושני של קביעת התוצאות ועבור המשך משחק משמשת לקרבות בשלב הattack.

```
bool operator>(const Cube&) const;
bool operator==(const Cube&) const;
```

| Ctor | Cube(); (const Player&); |
|--|---|
| = אופרטור | bool operator<(const Cube&) const; |
| | bool operator>(const Cube&) const; |
| | bool operator==(const Cube&) const; |
| מציגה ומסתירה את הקובייה על פני ה gui. | <pre>void hide();</pre> |
| | <pre>void show();</pre> |
| השמה והוספה של חיילים עבור כיבוש של מדינות | <pre>void choosePic(int);</pre> |
| וטריטוריות. | <pre>void setValue(int);</pre> |
| | <pre>int getValue() const;void</pre> |
| | <pre>setBonusTroops(const int);</pre> |
| קביעת הנתיב של התמונה שאותה רוצים לטעון לפי | <pre>Void setpath(std::string)</pre> |
| ההגרלה. | |
| . id שומרת את השם של השחקן צבע סדר ו | <pre>void save(std::ofstream& file)</pre> |
| | <pre>void load(std::ifstrsceam& file)</pre> |
| טוענת את הפרמטרים של השחקן שנשמרו לקובץ . | |
| מציגה את הקלפים של השחקן על גבי ה gui. | <pre>void show()</pre> |
| ייי - מנקה את הקלפים של השחקן במידה והשתמש בהם (- | |
| , gui מה | <pre>void cleanCards(std::vector<card></card></pre> |
| .gui ing. | cards) |
| | , |
| | |
| | Members |
| מכיל פרטים אודות השחקן בהתאמה ./ | Jint id name color |
| מחזיק את הסדר של השחקן בתור | <pre>int order;</pre> |
| מחזיק את הקלפים של השחקן. | std::vector <card> cards;</card> |
| " ' ' | |
| מחזיק את החיילים של השחקן שמגיעים לו בהתאם | <pre>int bonusTroops;</pre> |
| לכיבישו והיבשות בהם הוא שולט. | |
| יל ב טו ווו בטוול בוום וווא טווט. | |

<u>: Player המחלקה</u>

| | Player(const std::string name = |
|---|--|
| | "default", const std::string color = |
| | "None", int order = 0, int id = -1); |
| | Player(const Player&); |
| וופרטור = | |
| | Player&); |
| תנקה את הקלפים של השחקן במידה והשתמש בהם – | <pre>void cleanCards(std::vector<card></card></pre> |
| על גבי ה gui. | - cards) |
| _ | |
| השמה והוספה של חיילים עבור כיבוש של מדינות | |
| טריטוריות. | <pre>void setBonusTroops(const int);</pre> |
| זוסיםה קלך לשחקןץ | Card addCard(); |
| . id אומרת את השם של השחקן צבע סדר ו | <pre>void save(std::ofstream& file)</pre> |
| | <pre>void load(std::ifstrsceam& file)-</pre> |
| . ווענת את הפרמטרים של השחקן שנשמרו לקובץ | |
| ציגה את הקלפים של השחקן על גבי ה gui. | <pre>void show()—</pre> |
| מנקה את הקלפים של השחקן במידה והשתמש בהם | |
| יַל גבי ה gui. | <pre>void cleanCards(std::vector<card></card></pre> |
| · · | cards) |
| | |
| | Members |
| יכיל פרטים אודות השחקן בהתאמה ./ | int id name color |
| מחזיק את הסדר של השחקן בתור | int order; |
| מחזיק את הקלפים של השחקן. | std::vector <card> cards;</card> |
| | |
| מחזיק את החיילים של השחקן שמגיעים לו בהתאם | <pre>int bonusTroops;</pre> |
| כיבישו והיבשות בהם הוא שולט. | |
| נל גבי ה gui. השמה והוספה של חיילים עבור כיבוש של מדינות טריטוריות. מוסיםה קלך לשחקןץ שומרת את השם של השחקן צבע סדר ו id . מציגה את הפרמטרים של השחקן שנשמרו לקובץ . מציגה את הקלפים של השחקן על גבי ה gui. מנקה את הקלפים של השחקן במידה והשתמש בהם gui. נל גבי ה gui. מחזיק את הסדר של השחקן בתור מחזיק את הקלפים של השחקן. | - cards) void addBonusTroops(const int); void setBonusTroops(const int); Card addCard(); void save(std::ofstream& file) void load(std::ifstrsceam& file)— void show()— void cleanCards(std::vector <card> cards) Members int id name color int order; std::vector<card> cards; int bonusTroops;</card></card> |

-מחלקה בעת שכובש טריטוריה מקבל קלף שחקן עבור המשתמש –בעת שכובש טריטוריה מקבל קלף <u>במחלקה: Card</u> בשווי של הקלף בחיילים מתנהל לפי סדרת פיבונאצי.

```
Card& operator=(const Card&);
bool operator==(const Card&) const;
bool operator!=(const Card&) const;
```

| מקדמת את הערך של מסחר ב3 קלפים לפי סדרת | <pre>static void nextBonus();</pre> |
|---|---|
| פיבונאצי. | |
| מחזירה את הערך הנוכחי של מספר ב3 קלפים | <pre>static int getBonus();</pre> |
| ctors | <pre>Card(cardType cardType =</pre> |
| | <pre>cardType::Default);</pre> |
| | <pre>Card(const Card&);</pre> |
| סוגי הקלפים הקיימים – כדי לבצע החלפה צריך 3 זוגות | <pre>enum cardType {</pre> |
| זהים או 3 שונים כאשר ג'וקר מתאים לכולם . | Joker, Solider, Cannon, |
| γ | Horse, Default }; |
| | |
| מבצע את ההגררלה של הקלפים מבין הסוגים הקיימים | <pre>void setRand();</pre> |
| | void |
| | <pre>setButton(QPushButton*);</pre> |
| . שמתאים לו Gui קישור בין קלף לבין לחצן ב | void |
| | <pre>setButton(QPushButton*);</pre> |
| . guiעדכון של קלפי המשחק של השחקן | <pre>void Card::update()</pre> |
| שמירה וטעינה של סוג הקלף לקובץ. | <pre>void save(std::ofstream&</pre> |
| 1 1 11 | file) const; |

| | <pre>void load(std::ifstream& file);</pre> |
|---|---|
| מנקה את התצוגה של הקלף מהgui. | <pre>void clean();</pre> |
| מחזיר את סוג הקלף | <pre>int getType() const;</pre> |
| מתודה המנהלת את הלחיצה של המתמש על פני הקלף- במידה וקיים קלף בידי השחקן אם הוא לוחץ אליו מציגה אותו עם צל ולחיצה נוספת מבטלת את הצל. syntax Slots- של הוg שמשמעותו-זו המתודה שמטפלת ב event בעת לחיצה –במקרה הזה המתודה מתקרת ללחצן שמקושר לקלף שעל גביו מוצגת תמונה של קלף. | <pre>public slots:- void click();</pre> |
| | Members |
| שלושת הערכים האחרונים בשווי הקלף לפי כמות חיילים – כאמור החלטנו שהשווי של מסחר ב3 קלפים ילך ויתקדם לפי סדרת פיבונאצי . | <pre>static int bonus1; static int bonus2; static int bonus3;</pre> |

STAGE המחלקה – מחלקה אבסטרקטית המכילה מתודות הדרושות ליישום שלבי המשחק.

| מאתחל את הדגלים nexten וenteren אשר מתפעלים את | Stage(Ui::my_riskClass&, |
|---|---|
| לחצן ה enter שתפקידו לבצע אישור של פעולות בשלבי | Player& player, Move& |
| המשחק וכפתור ה next שמאפשר לעבור בין שלבי | moveUi); |
| . המשחק | |
| מבצע פעולות המקדימות את השלב כמו חישבוים (חישוב | <pre>virtual void preAction()=0</pre> |
| gui מספר החיילים המגיע למשל ב draft) והצגה על פני ה | |
| ואתחול משתנים | |
| ביצוע כל הפעולות הקשורות בשלב אותו מבצעים | <pre>virtual void action()=0;</pre> |
| מבצע בדיקות האם מתקיימים התנאים שעבורם ניתן | virtual void |
| . Gui לעבור לשלב הבא ת מציג נתונים על גבי ה | afterAction()=0; |
| מציג את מספר החיילים שאותם זכאי המשתמש למקם על | virtual void |
| גבי המדינות שבבעלותו וכמו כן מספר החיילים שאיתם הוא | <pre>showTroops();</pre> |
| בוחר לתקוף בשלב התקיפה . | |
| – showtroops מאפשר שליטה על הטקסט שמוצג ב | <pre>virtual void plus(int sel =</pre> |
| הוספת חיילים למקום מסוים- תקיפה עם יותר חיילים . | 0); |
| בדומה לvoid plus רק שמאפשר להפחית . | virtual void |
| | minus(int sel = 0); |
| Getters למצב הלחצנים. | <pre>bool getNexEn();</pre> |
| | <pre>bool getEnterEn();</pre> |
| tters ללחצנים | void setNextEn(const bool |
| | en); |
| | <pre>void setEnterEn(const bool</pre> |
| | en); |
| | void |
| Setter שמבצע השמה לרפרנס של השחקן הנוכחי | setPlayer(Player&); |
| שמשחק . | |
| מחזיר את מספר חיילי התגבור שלהם זכאי השחקן- | <pre>int getBonusTroops(int sel = 0);</pre> |
| שימושי בשלב ה draft . | • • |
| בודק אם מדינה שנלחצה שייך לשחקן הנוכחי שמשחק – | <pre>bool isMine(int sel = 0);</pre> |
| על סמך כך נקבעות החלטות בשלבים שונים במשחק | Mawlaga |
| | Members |
| רפרנס לשחקן הנוכחי שמשחק | Player& player; |
| | |
| next מצב כפתור ה | bool nextEn; |
| enter מצב כפתור ה | bool EnterEn; |
| נובב ככונוו ווווסאוים | , |

stagen יורשת יורשת <u>PlaceMap</u>

אחראית למימוש השלב הראשון של כיבוש החלקים במפה ע"י המשתתפים.

| gui אחראית להצגה של נתונים של השלב הראשון על גבי ה | <pre>void preAction()</pre> |
|--|-------------------------------|
| אחראית למימוש השלב הראשון. | <pre>void action()</pre> |
| מבצעת השמה לנתונים על פני הgui לאחר ביצוע השלב. | <pre>void afterAction()</pre> |
| | Members |
| משמש לספור כמה חיילים חילקנו עד עכשיו . | <pre>int& counter;</pre> |
| | |

stagen יורשת <u>PlaceTroops</u>

אחראית לביצוע השלב השני באתחול המשחק- מיקום כל החיילים שחולקו לשחקנים בשלב הראשוני על פני הלוח.

| אחראית למימוש השלב השני. | <pre>void action()</pre> |
|---|-------------------------------|
| אחראית להצגה של נתונים של השלב השני על גבי ה gui. | <pre>void preAction()</pre> |
| - | <pre>void afterAction()</pre> |
| .gui מציג מידע על על גבי ה | <pre>void showInfo();</pre> |
| | Members |
| משמש לספור כמה חיילים חילקנו עד עכשיו . | <pre>int& counter;</pre> |
| | |

<u>המחלקה draft:</u> יורשת מstage-מממשת את שלב ה draft –בשלב זה השחקן ממקמם את החיילים שהוא מקבל בתחילת כל תור במדינות שבבעלתו – מספר המדינות נקבע על סמך הכיבושים שלו ומספר היבשות שהוא כובש.

| drafta אחראית למימוש השלב של | <pre>void action()</pre> |
|---|---|
| על גבי draft אחראית להצגה של נתונים של השלב | <pre>void preAction()</pre> |
| ה gui ובנוסף לחישוב מספר החיילים המגיע לשחקן | |
| בתחילת התור והצגת הקלפים שבבעלותו. | |
| .gui מציג מידע על על גבי ה | <pre>void afterAction()</pre> |
| .gui מציג מידע על על גבי ה | <pre>voidshowBonusTroops(std::string)</pre> |
| מחשבת את מספר החיילים המגיעים לשחקן בהתאם | Land::landInfo calcTroops(); |
| לכיבשיו והקלפים שבידיו- במידה ובידיו 5 קלפים חייב | |
| לסחור בהם . | |
| | |
| מנהלת את הלחיצות על כפתור ה+ למיקום חיילם | <pre>void plus(int sel = 0);</pre> |
| בטריטוריה מסויימת. | |
| מנהלת את הלחיצות על כפתור ה- למיקום חיילם | <pre>void minus(int sel = 0);</pre> |
| בטריטוריה מסויימת. | |
| אחראית להחלפת קלפים של שחקן תמורת חיילים | <pre>void convert();</pre> |
| לתגבור | |
| | members |
| רפרנס לקלפים של הgui. | std::vector <card>& cards;</card> |
| פויינטר ליבשות של המשחק. | std::vector <land*> lands;</land*> |

<u>המחלקה attack :</u> יורשת מstage -אחראית על מימוש שלב ההתקפה של שחקן – בשלב זה השחקן יכול לכבוש מדינות הסמוכות לטריטריות שבבעלתו זאת ע"י לחיצה על מדינה שלו ומדינה שכנה והתמודדות בקרב של הטלת קוביות- השחקן יכול להטיל עד 3 המותקף יכול להגן עד 2.

| attackאחראית למימוש השלב של ה | <pre>void action()</pre> |
|---|------------------------------------|
| על גבי draft אחראית להצגה של נתונים של השלב | <pre>void preAction()</pre> |
| ה gui ובנוסף לחישוב מספר החיילים המגיע לשחקן | |
| בתחילת התור והצגת הקלפים שבבעלותו. | |
| מציג מידע על על גבי ה gui. | <pre>void afterAction()</pre> |
| gui מציג מידע על על גבי ה | <pre>void plus(int sel = 0)</pre> |
| מחשבת את מספר החיילים המגיעים לשחקן בהתאם | <pre>void minus(int sel = 0)</pre> |
| לכיבשיו והקלפים שבידיו- במידה ובידיו 5 קלפים חייב | |
| לסחור בהם . | |
| | |
| | Members |
| מסמל אם מגיע card חדש או לא | <pre>int newCard;</pre> |
| פוינטר לקלף במקרה של כיבוש מדינה. | Card* card; |
| פוינטר לקובץ אודיו שעושה צליל של פיצוף כאשר יש | QMediaPlayer* boom; |
| . כיבוש של מדינה | |

המחלקה reinforcement יורשת מstage-אחראית למימוש שלב התגבור בו המשתמש יכול להעביר חיילים מטריטוריה מסוימת שלו לטריטוריה אחרת שלו בתנאי שיש מסלול רציף שמחבר אותם. בדומה לשלבי המשחק הקודמים ממש את המתודות:

```
void action();
void preAction();
void afterAction();
```

המחלקה land במשחק יש 6 יבשות של המשחק במשחק יש 6 יבשות

| אובייקט המחזיר מידע לגבי הבונוס המגיע למי שמחזיק ביבשת gui להצגה על גבי ה gui – במידה ושחקן כבש את כל היבשת מציג- כמה שווה היבשת והסיבות שבגינן הוא זכאי למספר החיילים שקיגבל. | <pre>struct landInfo { int bonus; std::string reason; };</pre> |
|---|--|
| השמה של יבשת ליבשת . מאתחל land בשם ובland | <pre>Land& operator=(Land&) Land(std::vector<std::string>,</std::string></pre> |
| מאונות bilal בשם ובסוחומוהו | Land::landInfo) |
| מחשבת את מספר החיילים אשר להם זכאי שחקן | Land::landInfo |
| עבור שליטה במדינות ביבשת. | calcTroops(Player&) |
| שמירה וטעינה של פרמטרים הקשורים למחלקה | <pre>void save(std::ofstream&</pre> |
| לקובץ –ריצה בלולאה על כל המדינות ששיכות ליבשת | file); |
| ושימוש בפונקציית השמירה והטעינה של מדינה . | <pre>void load(std::ifstream& file,</pre> |
| · | <pre>std::map<std::string, player="">& playersMap);</std::string,></pre> |
| | Members |
| מחזיק את המדינות הנמצאות על היבשת. | std::vector <country></country> |
| | countries; |

| אינוסטרקטור וקופי קונסקרטור (Country(const std::string, Player player = Player()) (Country(const Country &); (Country(); (Cou | | |
|--|--|---|
| Player player = Player()) Country(const Country &); Country(s); Country(s) operator=(Country&); Country(s); Country(s); Country(s); Country(s); Country(s); Country(s); Country(s); Country(s); //setters player etlayer&); void setButton(pPushButton*); void setButton(pPushButton*); void setButton(pPushButton*); void setNehibors(std::vector <country*>); //getters Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; int getTroops(); bool isNear(std::string) const; bool isNear(std::string dent and the first a</country*> | קונסטרקטור וקופי קונסקרטור | |
| Country(const Country&); Country& operator=(Country&); Country(); //setters //setters void setOwner(const Player&); static void setUI(Ui::my_riskClass*); void setButton(PushButton*); void setNehibors(std::vectorcCountry*>); //getters Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; int getTroops(); bool isNear(std::string) const; int getTroops(); bool isNear(std::string) const; bool isNear(std::string) const; country*:validPath(const std::string dest, std::string sool>& seen) woid save(std::ofstream& file) const; void load(std::ifstream& file); // woid init(); // Members: static Country* chosen; static Ui::my_riskClass* ui; int troops; std::string name; std::vector <country*> nehibors; Player owner; Player owner;</country*> | | Country(const std::string, |
| Country& operator=(Country&); Country(); (Country(); (Country()); (Country(| | |
| Country(); //setters איים המחלקה – שכנים , cenir שאליו המדינה מחלקה – שכנים , cenir שאליו המדינה משוכית , eiviu setWesters setWesters setNehibors(std::vector <country*>); //getters Amazer setNehibors(std::vector<country*>); //getters setNehibors(std::string const; setIng getName() const; setIng getName()</country*></country*></country*></country*></country*></country*></country*></country*></country*></country*> | | |
| אריו המחלקה —שכנים , כפתור void setters void settomer (const Player&); void settovoid setUI(Ui::my_riskClass*); void setButton(QPushButton*); void setButton(QPushButton*); void setWeltion*(QPushButton*); int getTroops(); bool isNear(std::string dent and the price of the | | |
| void setOwner(const Player&); static void setUl(Ui::my_riskClass*); void setButton(QPushButton*); void setButton(QPushButton*); void setButton(QPushButton*); void setButton(QPushButton*); void setButton(QPushButton*); void setButton(QPushButton*); void setNehibors(std::vector <country*>); //getters Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; bool isNear(std::string) const; and control const bool country::validPath(const std::string dest, std::map<std::string,bool>& seen) bool country::validPath(const std::string dest, std::map<std::string,bool>& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); members: static Country* chosen; static Country* chosen; static Country* chosen; static Country* chosen; static Country chosen; static Country chosen; static Country chosen; static String name; std::stering name; std::stector player owner; Player owner;</std::string,bool></std::string,bool></country*> | | |
| static void setUI(Ui::my_riskClass*); void setButton(QPushButton*); void setNehibors(std::vector <country*>); //getters Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; int getTroops(); bool isNear(std::string) const; and and a circl hour him him him him him him him him him him</country*> | Setters למשתני המחלקה –שכנים ,כפתור | • • |
| אינים | שאליו המדינה משוכית ,פוינטר לgui, שחקו | |
| איסים | | |
| setNehibors(std::vector <country*>); //getters Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; bool country::validPath(const std::string dest, std::map<std::string dest,="" std::map<std::string,bool="">& seen) country::validPath(const std::string dest, std::map<std::string,bool>& seen) public slots: void load(std::ifstream& file); public slots: void init(); public slots: void init(); Members: static Country* chosen; static Country* chosen; static Country* chosen; static Country* chosen; static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::string name; std::vector<country*> nehibors; Player owner; Player owner; pud car pud c</country*></std::string,bool></std::string></country*> | .0110 | \ * * * * * * * * * * * * * * * * * * * |
| //getters Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool country::validPath(const std::string dest, std::map <std::string dest,="" std::map<std::string,bool="">& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); Members: static Country* chosen; getiving dest, std::map<std::string hool="">& seen) public slots: void init(); bool country::validPath(const std::string dest, std::map<std::string hool="">& seen) public slots: void init(); bool const; std::string hy metric slots: void init(); public slots: void init(); bool const; std::string hy metric slots: void init(); bool country* chosen std::country* chosen; static Country* chosen; static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::string name; std::vector<country*> nehibors; Player owner; Player owner;</country*></std::string></std::string></std::string> | | |
| Player getOwner() const; std::string getName() const; int getTroops(); bool isNear(std::string) const; bool contry::validPath(const std::string dest, std::string dest, std::string dest, std::string, bool>& seen) coid save(std::ofstream& file) const; void load(std::ifstream& file); const; void load(std::ifstream& file); public slots: void init(); coid init(); coi | | |
| std::string getName() const; int getTroops(); dark detrice string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool isNear(std::string) const; bool country::validPath(const std::string dest, std::string dest, std::string dest, std::string, bool save(std::string, bool) seen) backtracking may void save(std::ofstream& file) const; void load(std::ifstream& file); void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); Members: static Country* chosen; getivin. metricin mannicin methali arru(nivia) equiva device arruin. int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner; Player owner;</country*> | getters למשתני המחלקה | |
| int getTroops(); מתודה לבדיקה אם מדינה סמוכה- על מנת during is Near(std::string) const; bool isNear(std::string) const; drum ha evid in the price of its is nearly const; bool isNear(std::string) const; which price of its | | |
| מתודה לבדיקה אם מדינה סמוכה- על מנת לדעת אם ניתן להעביר אליה חיילים או לתקוף לדעת אם ניתן להעביר אליה חיילים או לתקוף מסדר ראשון. אותה בשלבי המשחק השונים-בודק קרבה מסדר ראשון. בודק האם קיים מסלול רציף של מדינות של bool Country::validPath(const std::string dest, std::string dest, std::map <std::string,bool>& seen) bool Country::validPath(const std::string dest, std::string dest, std::map<std::string,bool>& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); Members: static Country* chosen; static Country* chosen; static Country* chosen2; static Country* chosen2; int troops; static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector<country*> nehibors; QPushButton* button; Player owner;</country*></std::string,bool></std::string,bool> | | |
| לדעת אם ניתן להעביר אליה חיילים או לתקוף אותה בשלבי המשחק השונים-בודק קרבה מסדר ראשון. מסדר ראשון. בודק האם קיים מסלול רציף של מדינות של acktracking בודק האם קיים מסלול רציף של מדינות של std::string dest, std::string,bool>& seen) באמצעות backtracking mame; באמצעות std::string,bool>& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); Members: static Country* chosen; static Country* chosen; static Country* chosen2; arru(חיילים) בין שתי מדינות. מידע(חיילים) בין שתי מדינות. מידע(חיילים של גבי המדינה static Ui::my_riskClass* ui; מידע(חיילים של גבי המדינה static Ui:my_riskClass* ui; מידע(חיילים של המדינה std::string name; בל::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | <pre>int getTroops();</pre> |
| אותה בשלבי המשחק השונים-בודק קרבה מסדר ראשון. מסדר ראשון. בודק האם קיים מסלול רציף של מדינות של chosen (std::string, bool) (save(std::string, bool) (seen) מעדכן את שתי המדינות האחרונות שנלחצו (void save(std::ofstream& file) (sonst; void load(std::ifstream& file); public slots: void init(); public slots: void init(); public slots: void init(); Members: attic Country* chosen; static Country* chosen; static Country* chosen2; attic Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | bool isNear(std::string) const; |
| מסדר ראשון. מסדר ראשון. בודק האם קיים מסלול רציף של מדינות של bool Country::validPath(const std::string dest, std::string,bool>& seen) באמצעות backtracking שמירה וטעינה של המשחק. באמצעות save(std::ofstream& file) const; void load(std::ifstream& file); מעדכן את שתי המדינות האחרונות שנלחצו public slots: void init(); שמוטי בול chosen והשיכות שלהם למששתמשים או אם הם eliיות. השייכות שלהם למששתמשים או אם הם eliיות. Members: באמציעם לשתי המדינות האחרונות שנלחצו שימושי בכל שלבי המשחק שרוצים להעביר מידע(חיילים) בין שתי מדינות. באמציעם של המדינות static Ui::my_riskClass* ui; באמציעם של המדינה int troops; באמציע של המדינה std::string name; באמציע של המדינה הפתור המשויך למדינה (PushButton* button; Player owner; | | |
| bool Country::validPath(const std::string dest, std::string dest, std::string dest, std::map <std::string,bool>& seen) can backtracking backtracking const; void save(std::ofstream& file) const; void load(std::ifstream& file); void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); Members: arzevud dumr natritin mannitin defnril mitter country* chosen; static Country* chosen; static Country* chosen2; static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector<country*> nehibors; QPushButton* button; Player owner;</country*></std::string,bool> | אותה בשלבי המשחק השונים-בודק קרבה | |
| std::string dest, std::maprostd::string,bool>& seen) std::maprostd::string,bool>& seen) std::maprostd::string,bool>& seen) woid save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); members: static Country* chosen; static Country* chosen; static Country* chosen2; pustic Country* chosen; static Country* chosen; static Country* chosen2; static Ui::my_riskClass* ui; int troops; std::string name; std::vector <country*> nehibors; opushButton* button; Player owner; Player owner; poid save(std::string,bool>& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); static Country* chosen; static Country* chosen; static Ui::my_riskClass* ui; int troops; std::string name; std::vector<country*> nehibors; opushButton* button; Player owner;</country*></country*> | מסדר ראשון. | |
| std::string dest, std::map <std::string,bool>& seen) std::map<std::string,bool>& seen) std::map<std::string,bool>& seen) std::map<std::string,bool>& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); members: static Country* chosen; static Country* chosen; static Country* chosen2; static Country* chosen2; static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector<country*> nehibors; QPushButton* button; Player owner; Player owner;</country*></std::string,bool></std::string,bool></std::string,bool></std::string,bool> | בודק האם קיים מסלול רציף של מדינות של | bool Country::validPath(const |
| אמירה וטעינה של המשחק. std::map <std::string,bool>& seen) באמצעות std::map<std::string,bool>& seen) void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); members: Members: - מצביעם לשתי המדינות האחרונות שנלחצו etinin. מידע(חיילים) בין שתי מדינות שנלחצו static Country* chosen; static Country* chosen; static Country* chosen2; מידע(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::string name; std::vector<country*> nehibors; QPushButton* button; Player owner;</country*></std::string,bool></std::string,bool> | | |
| void save(std::ofstream& file) const; void load(std::ifstream& file); public slots: void init(); public slots: void init(); Members: Members: static Country* chosen; static Country* chosen2; arry(חיילים) בין שתי מדינות. static Country* chosen; static Country* chosen2; int troops; int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner; Player owner;</country*> | <u> </u> | <pre>std::map<std::string,bool>& seen)</std::string,bool></pre> |
| void load(std::ifstream& file); מעדכן את שתי המדינות האחרונות שנלחצו public slots: void init(); השייכות שלהם למששתמשים או אם הם elinit(); Members: מצביעם לשתי המדינות האחרונות שנלחצו – elinit מצביעם לשתי המדינות האחרונות שנלחצו – static Country* chosen; static Country* chosen2; מידע(חיילים) בין שתי מדינות. מידע(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; int troops; int troops; השם של המדינה std::string name; std::vector<country*> nehibors;</country*> QPushButton* button; Player owner; | | void save(std::ofstream& file) const: |
| מעדכן את שתי המדינות האחרונות שנלחצו chosen ו chosen ו chosen (מדפיס נתונים לגבי void init(); שלות. השייכות שלהם למששתמשים או אם הם (מניות. השרונות שנלחצו אם הם (מניות. השרונות שנלחצו שימושי בכל שלבי המשחק שרוצים להעביר (מידע(חיילים) בין שתי מדינות. מידע(חיילים) בין שתי מדינות. מספר החיילים שעל גבי המדינה (מודע troops; static Vi::my_riskClass* ui; and troops; std::string name; std::string name; std::vector <country*> nehibors; OPushButton* button; Player owner;</country*> | שניו זו וטעינוו של זומשווק. | |
| <pre>void init(); reflection of the proof of the proof</pre> | | Total Toda(Sed: Trochedina 1712) |
| <pre>void init(); reflection of the proof of the proof</pre> | מעדכו את שתי המדינות האחרונות שנלחצו | public slots: |
| members: Members: Members: a static Country* chosen; static Country* chosen; static Country* chosen; static Country* chosen2; arru(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; and int troops; int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | |
| פנויות. Members: a varium and purium action described by the process of the pr | <u> </u> | (// |
| אפשים לשתי המדינות האחרונות שנלחצו – static Country* chosen; מצביעם לשתי המדינות האחרונות שנלחצו – static Country* chosen; מידע(חיילים) בין שתי מדינות. מידע(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | |
| static Country* chosen; מצביעם לשתי המדינות האחרונות שנלחצו – static Country* chosen; שימושי בכל שלבי המשחק שרוצים להעביר מידע(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | פנויות. | Mandago |
| אימושי בכל שלבי המשחק שרוצים להעביר static Country* chosen2; מידע(חיילים) בין שתי מדינות. מידע(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; int troops; std::string name; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | |
| מידע(חיילים) בין שתי מדינות. מידע(חיילים) בין שתי מדינות. static Ui::my_riskClass* ui; int troops; std::string name; std::vector <country*> nehibors; Player owner;</country*> | | |
| פוינטר לius static Ui::my_riskClass* ui; מספר החיילים שעל גבי המדינה int troops; std::string name; std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | <u>.</u> | static Country* cnosen2; |
| int troops; מספר החיילים שעל גבי המדינה std::string name; השם של המדינה std::vector <country*> nehibors; מספר המשויך למדינה (PushButton* button; Player owner;</country*> | מידע(חיילים) בין שתי מדינות. | |
| int troops; מספר החיילים שעל גבי המדינה std::string name; השם של המדינה std::vector <country*> nehibors; מספר המשויך למדינה (PushButton* button; הבעלים של המדינה (Player owner;</country*> | פוינטר לgui הראשי. | <pre>static Ui::my_riskClass* ui;</pre> |
| std::string name; השם של המדינה std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | <pre>int troops;</pre> |
| std::vector <country*> nehibors; QPushButton* button; Player owner;</country*> | | <pre>std::string name;</pre> |
| הבעלים של המדינה –נדרש רק לדעת Player owner; | | <pre>std::vector<country*> nehibors;</country*></pre> |
| הבעלים של המדינה –נדרש רק לדעת Player owner; | הכפתור המשויך למדינה | QPushButton* button; |
| | | Player owner; |
| | | |

המחלקה move :

| סטרים לאתחול מספר מינימלי ומקסימלי של שחקנים להעברה | <pre>void setMax(int); void setMin(int);</pre> | | |
|--|--|--|--|
| | , ,, | | |
| קבלת מספר חיילים להעברה | <pre>int getBonusTroops(int</pre> | | |
| · | sel = 0); | | |
| אתחול יעד ומקור להעברת החיילים | <pre>void setFrom(Country*);</pre> | | |
| | <pre>void setTo(Country*);</pre> | | |
| | Members | | |
| מתודות ללחצנים של פלוס ומינוס לקביעת מספר | <pre>void plus(int sel</pre> | | |
| החיילים | = 0); | | |
| B 7 1111 | <pre>void minus(int sel = 0);</pre> | | |
| מתודת אישור | <pre>void end();</pre> | | |
| ממשק משתמש אבא | QWidget* father; | | |
| ממשק משתמש שלו. | Ui::Move ui; | | |

על מנת ליצור בה כפתור בgui שיתקשר לתפריט. QWidget יורשת מ יורשת מ' Qwidget

| קונסטרקטור וקופי קונסקרטור | //ctor | | |
|--|---|--|--|
| | menu(QWidget * parent | | |
| | = Q_NULLPTR); | | |
| | | | |
| מתודת לחצן לאישור הנתונים ותחילת משחק | <pre>void end();</pre> | | |
| | | | |
| טיפול בבחירת מספר שחקנים | <pre>void spin(int);</pre> | | |
| הטלת קוביות | <pre>void cube();</pre> | | |
| שמירה וטעינה של המשחק. | <pre>void save(std::ofstream&</pre> | | |
| Ψ | file) const; | | |
| | <pre>void load(std::ifstream&</pre> | | |
| | file); | | |
| | | | |
| . dialog טעינת קובץ משחק באמצעות | <pre>void LOAD();</pre> | | |
| טעינת קובץ סטטיסטיקות ופתיחת ממשק משתמש | <pre>void loadStats();</pre> | | |
| לסטטיסטיקות | | | |
| טעינת קובץ ברירת מחדל לסטטיסטיקות מתעדכן | <pre>void defaultStats();</pre> | | |
| באופן רציף בכל משחק . | | | |
| | Members: | | |
| ממשתק משתמש. | Ui::menu ui; | | |
| אוביקט המשחק. | risk w; | | |
| ממשק משתמש סטטיסטיקות. | statsGuiMain s; | | |
| קוביות לקביעת סדר | Cube cubes[4]; | | |
| שחקנים לאיתחול | Player players[4]; | | |
| היסטוגרמה למיון השחקנים על סמך הטלת הקוביות- | Player* hist[6][4]; | | |
| עמודות זה מספרים מ1-6 שורות –הפוינטרים | | | |
| לשחקנים. | | | |
| מספר השחקנים הנבחר | <pre>int numberOfPlayers;</pre> | | |
| דגל – מציין אם נעשתה הטלה-חייב לבצע הטלה על | bool toss; | | |
| מנת להתחיל את המשחק | | | |
| דגל מצב ברירת מחדל | <pre>bool defaultStats_;</pre> | | |
| t e e e e e e e e e e e e e e e e e e e | | | |

בה gui שיתקשר למשחק. QMainWindow יורשת מ יורשת מ יורשת מ

| 2.022.020.2 | //ctor | | |
|---|---|--|--|
| קונסטרקטור | risk(int numberOfPlayers, | | |
| | QWidget *parent = 0); | | |
| | QWIAGEC PARENC = 0); | | |
| מספר שחקנים | <pre>char get_number_of_player();</pre> | | |
| | googlesses _co_p_s (// | | |
| NOON WORKER | <pre>void set_number_of_player(char num);</pre> | | |
| מספר שחקנים | void set_number_or_prayer (char num); | | |
| | | | |
| הטלת קוביות | <pre>void cube();</pre> | | |
| שמירה וטעינה של המשחק. | <pre>void save(std::ofstream& file) const;</pre> | | |
| | <pre>void load(std::ifstream& file);</pre> | | |
| .== | void incPlayon(): | | |
| קידום שחקן | <pre>void incPlayer();</pre> | | |
| איתחול חיילים לשחקנים | <pre>void initTroops();</pre> | | |
| הצגת נתוני החיילים במסך | <pre>void showTroops();</pre> | | |
| queue פעולות על הקונטיינר | <pre>bool empty() const;</pre> | | |
| | <pre>void pop();</pre> | | |
| | <pre>void push(const Player&);</pre> | | |
| טיפול בשמירת וטעינת משחק – מתודות לגוי | <pre>void save(std::ofstream& file);</pre> | | |
| ופינמיות | <pre>void load(std::ifstream& file);</pre> | | |
| | void LOAD(). | | |
| | <pre>void LOAD(); void SAVE();</pre> | | |
| WASHING THINGS BY BITHSOUSING BRITIS | PlayerStats* loadStats(std::string | | |
| טעינת סטטיסטיקות עם ממשק משתמש | fileName = "Resources/stats.csv"); | | |
| | void loadStatsGui(); | | |
| (refresh) הצגת נתונים מחדש על המסך | <pre>void showInfo();</pre> | | |
| ווצגונ נונונים מווו פיעי וומסן (וופפוופו) | <pre>void updateStage();</pre> | | |
| מתודת לחיצת לחצן של קל, | <pre>void cardClick(int sel = 0);</pre> | | |
| פתיחת גוי להצגת הודעה | <pre>void showMess(std::string);</pre> | | |
| אישור פעולה | void ENTER(); | | |
| | void SAVE_ALL(); | | |
| שמירת משחק + סטטיסטיקה | | | |
| שמירת סטטיסטיקה | <pre>void saveStats();</pre> | | |
| יציאה מהמשחק ושמירת סטטיסטקות | <pre>void EXIT();</pre> | | |
| מעבר מצב משחק | <pre>void nextState();</pre> | | |
| | Members: | | |
| ממשתק משתמש. | Ui::my_riskClass ui; | | |
| תור שחקנים תור שחקנים | std::queue <player> queue;</player> | | |
| יבשות | Land SouthAmerica, NorthAmerica, | | |
| | Europe, Asia, Australia, Africa; | | |
| כפתורים לכל יבשת -> ממשק משתמש | std::vector <qpushbutton*></qpushbutton*> | | |
| · | SA_buttons; | | |
| | <pre>std::vector<qpushbutton*></qpushbutton*></pre> | | |
| | NA_buttons; | | |
| | std::vector <qpushbutton*></qpushbutton*> | | |
| | EU_buttons; | | |
| | std::vector <qpushbutton*></qpushbutton*> | | |
| | AS_buttons; | | |
| | std::vector <qpushbutton*></qpushbutton*> | | |
| | <pre>AU_buttons; std::vector<qpushbutton*></qpushbutton*></pre> | | |
| | AF_buttons; | | |
| מספר שחקנים | char numberOfPlayers; | | |
| • | int numberOfCountries; | | |
| מספר מדינות | THE HAMBEL OF COURTE TES, | | |

| מספר חיילים לכל שחקן | <pre>int troopsPerPlayer;</pre> | | |
|-------------------------------------|---|--|--|
| מונה מספר חיילים שנשלחו-משמש בשני | int counter; | | |
| השלבים הראשונים במשחק מציג כמה נשאר | | | |
| עוד למקם על פני ה Gui. | | | |
| דגל קלף חדש | bool newCard; | | |
| מחרוזת מצב משחק נוכחי | | | |
| | <pre>std::string stage;</pre> | | |
| קלפים שמיוחסים לממש משתמש | <pre>std::vector<card> cards;</card></pre> | | |
| מתודת משחק משתמש ללחיצת כרטיס | <pre>void cardClick(int sel = 0);</pre> | | |
| אובייקט סטטיסטיקות | PlayerStats stats; | | |
| זמן תחילת משחק- לשמירה בסטטיסטקות. | <pre>time_t startTime;</pre> | | |
| אוביקט ממשק משתמש של הודעה | Mess mess; | | |
| ממפה מחרוזת של שחקן לשחקן | <pre>std::map<std::string, player=""></std::string,></pre> | | |
| | playersMap; | | |
| אוביקט ממשק משתמש להעברת שחקנים בין | Move moveUi; | | |
| מדינות | | | |
| פוינטר לאוביקט מצב נוכחי | Stage* currentStage; | | |
| ממפה מחרוזת של מצב למחרוזת של מצב | <pre>std::map<std::string, std::string=""></std::string,></pre> | | |
| שבא לפניה- לצורך אתחול נתונים. | stage2preStage; | | |
| · | | | |

<u>המחלקה PlayerStats</u>מכילה מטודות כלליות של סטטיסטקות על השחקנים – המידע על ה שחקנים שמור ב container templet מסוג vector עליו יפורט בהמשך. הacontainer templet מכיל אובייקטים מסוג PlayerStatWrap המכיל מידע סטטיסטי על כל השחקנים שאי פעם שיחקו.

| קונסטרקטור | <pre>//ctor PlayerStats();</pre> | | |
|---------------------------------------|---|--|--|
| | . 14,0.0000(7,5 | | |
| גטרים – קבלת מספר מקסימלי של : זמן | PlayerStats::Info <int></int> | | |
| משחק,נצחונות,הפסדים,מספר משחקים | <pre>getMaxWins();</pre> | | |
| | PlayerStats::Info <int></int> | | |
| | <pre>getMaxLooses();</pre> | | |
| | PlayerStats::Info <int></int> | | |
| | <pre>getMaxTimePlayed(); PlayerStats::Info<int></int></pre> | | |
| | <pre>getMaxGamesPlayed();</pre> | | |
| מחרוזת של זמן ממוצע של משחק | std::string getAvgTime(); | | |
| בוווו ווונ פוי ומן ממובע פוי מפווון | Seattle Ing Seattle (/) | | |
| טעינה ושמירה של סטטיסטיקות | void | | |
| · | <pre>loadStats(std::ifstream& file);</pre> | | |
| | <pre>void saveStats(std::ofstream&</pre> | | |
| | file); | | |
| שמירה וטעינה של המשחק. | <pre>void save(std::ofstream& file) const;</pre> | | |
| | <pre>void load(std::ifstream& file);</pre> | | |
| הוספת סטטיסטיקות של שחקן על המעטפת | <pre>void addToStats(const</pre> | | |
| שלו | PlayerStatWrap&); | | |
| ניקוי סטטיסטיקות | <pre>void clear();</pre> | | |
| גודל של מספר שחקנים ששמורים | <pre>int size();</pre> | | |
| בסטטיסטיקה | | | |
| | | | |
| | Members: | | |
| Vector אוביקט של הקונטיינר האישי שלנו | Vector <playerstatwrap> players;</playerstatwrap> | | |

. המחלקה מכילה מידע סטטיסטי על שחקנים במשחק. PlayerStatWrap:

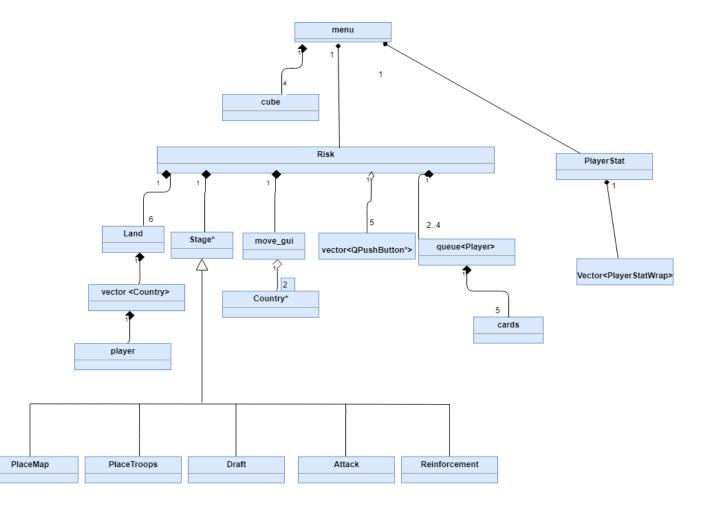
| קונסטרקטורים | //ctor | | |
|--|---|--|--|
| | PlayerStatWrap(const PlayerStatWrap& src); PlayerStatWrap(const Player& player,int win, int elapsed_seconds);PlayerStatWrap();; | | |
| קלאס זה חבר של קלאס הסטטיסטיקה של השחקן – הם באים ביחד. | friend class PlayerStats; | | |
| שמירה וטעינה של סטטיסטיקות האוביקט לקובץ ומקובץ | <pre>void loadStats(std::ifstream& file); void saveStats(std::ofstream& file);</pre> | | |
| אופרטורים | PlayerStatWrap& operator+(const PlayerStatWrap& src); PlayerStatWrap& operator=(const PlayerStatWrap& src); | | |
| | Members: | | |
| זמן ששיחק | <pre>int timePlayed;</pre> | | |
| מספר משחקים ששיחק | <pre>int gamesPlayed;</pre> | | |
| מספר נצחונות | int wins; | | |
| מספר הפסדים | int looses; | | |
| תז | int id; | | |
| שם | std::string name; | | |

<u>המחלקה : Vector</u> מממשת את מבנה הנתונים וקטור בו אנו משתמשים לשמור מידע סטטיסטי על המשתמשים-מחלקה template .בחרנו לממש מבנה זה כיוון שמספר השחקנים הנטען אינו יודע ווקטור הוא מבנה נוח בוא אפשר להוסיף באופן דינמי שאינו ידוע מראש כמות פריטים מבוקשת. \

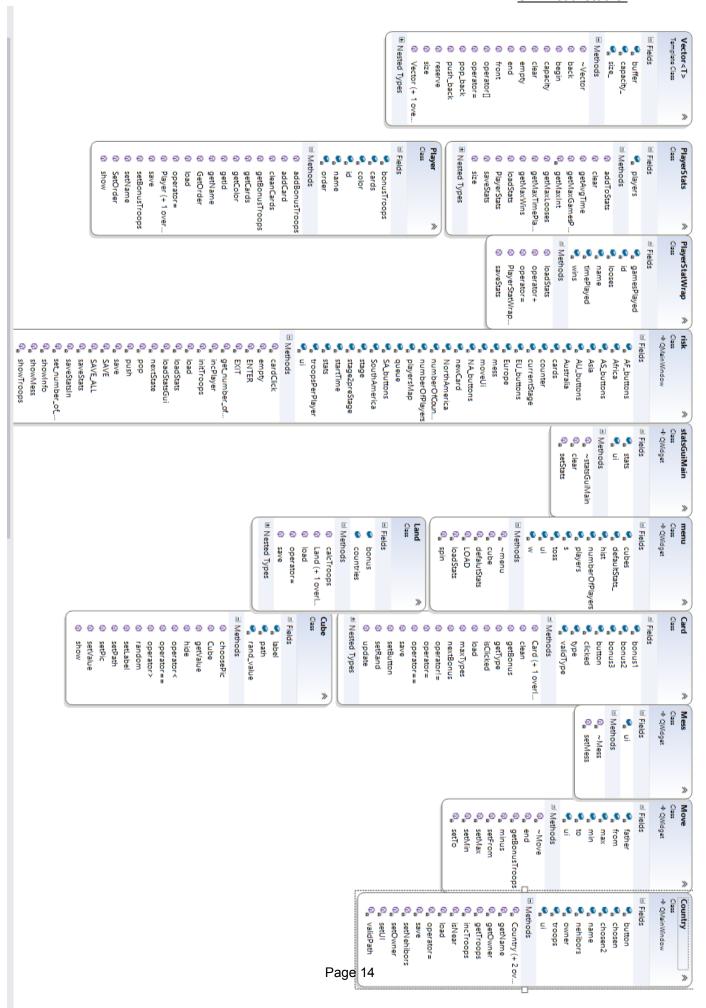
| Ctor | <pre>Vector();</pre> | | |
|--|--|--|--|
| . iterator ייצוג של | <pre>typedef T * iterator;</pre> | | |
| Copy ctor | <pre>Vector(const Vector<t>& v);</t></pre> | | |
| הגודל המוקצה של הווקטור –כל פעם שמתמלא מכפיל | <pre>unsigned int capacity()</pre> | | |
| . את עצמו פי 2 בגודל | const; | | |
| כמו מאוכלס בתוך הווקטור | <pre>unsigned int size()</pre> | | |
| | const; | | |
| דגל האומר האם הווקטור ריק | <pre>bool empty()</pre> | | |
| | const; | | |
| מחזיר את האיבר הראשון בווקטור | T& front(); | | |
| מחזיר את האיבר האחרון בוווקטור | T& back(); | | |
| מחזיר איטרטורים לתחילה ולסוף של הווקטור | <pre>iterator begin();</pre> | | |
| ' ' | <pre>iterator end();</pre> | | |
| מכניס איבר לווקטור | <pre>void push_back(const T& value);</pre> | | |
| מוציא איבר מהווקטור | <pre>void pop_back();</pre> | | |
| מקצה מקום בתוך הווקטור מראש | void reserve(unsigned int | | |
| | capacity); | | |
| אופרטור [] | T & operator[](unsigned int | | |
| | index); | | |
| מנקה את הווקטור | <pre>void clear();</pre> | | |

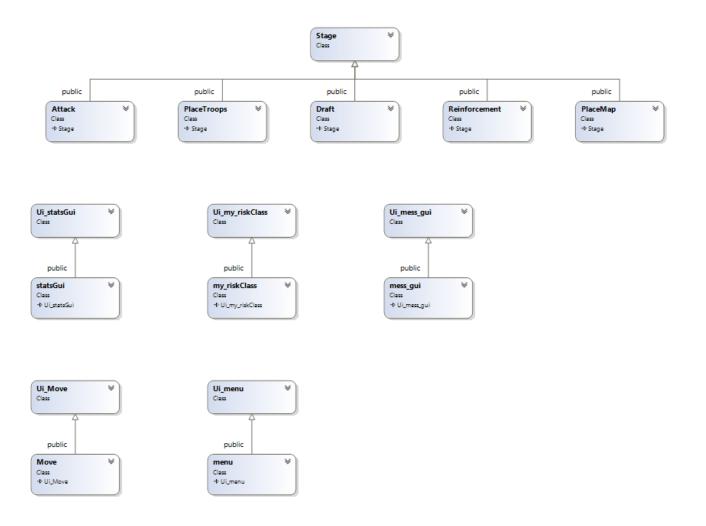
| = אופרטור | <pre>ector<t> & operator = (const Vector<t> &);</t></t></pre> | | |
|---|---|--|--|
| | members | | |
| מספר האיברים המוקצים בעלי מידע בתוך הווקטור | <pre>unsigned int size_;</pre> | | |
| הגודל שאליו מוקצה הווקטור | <pre>unsigned int capacity_;</pre> | | |
| המקום בו מאחסן את המידע. | T* buffer; | | |

יחסי הגומלין בין המחלקות:



Uml visual studio:





<u>מבני הנתונים בהם השתמשנו</u>:

- Map.1

stage על מנת לקשר בין ה std::map<std::string, std::string>1. הנוכחי ל stage הקודם לצורך טעינת משתנים ותצוגה על ה gui הנוכחי

std::map<std::string, Player>2.

playersMap –מקשר בין שם השחקן לבין אובייקט של שחקן בעת תהליך הseralization בטעינה מהקובץ.

s std::map<std::string, Country*> m3-משמש על מנת לקשר בין שם של ארץ לבין -s std::map<std::string, Country*> m3 הכפתור המקושר אליה- כך ניתן בקלות רבה יותר לקשר כל מדינה אל השכנות שלה- ע"י התאמת אינדקסים של שמות באופן אלגנטי שיקשרו כל מדינה ללחצנים המתאימים של השכנות שלה.

באופן כללי בחרנו להשתמש במבנה נתונים זה כי זמן החיפוש בו הוא מהיר , הוא נוח בקישור של מידע לפריטים במקום להתעסק עם אינדקסים חסרי משמעות מתעסקים עם שמות אינטואיטיביים – דבר אשר הופך את העבודה למאוד אלגנטית וברורה.

-queue.2

;std::queue<Player> queue -- השתמשנו במבנה נתונים זה על מנת לנהל את התורות של השחקנים במשחק מבנה זה נוח במיוחד לשימוש על מנת לנהל תור של fifo כיוון שניתן בקלות רבה להעביר מידע מתחילת התור לסוף התור כפי נדרש לעשות בניהול תור שחקנים.

vector.3 –השתמשנו במבנה נתונים זה על מנת להחזיק את קלפי המשחק שמספר משתנה דימנית וכמו כן להחזיק את השכנים של כל מדינה שגם מספרם הוא משתנה . השימוש בווקטור היה כיוון שהוא מבנה דינמי שמאפשר להרחיבו ולגשת לנתונים בו בקלות רבה. מבנה זה גם מומש לצרוך טעינת הסטטיסטיקות של השחקנים כיוון שאיננו יודעים כמה שחקנים הולכים לטעון.

:Seralizition

בחלק זה יצרנו קובץ שאליו אנחנו שומרים את נתוני המשחק . בחרנו לתת לקובץ את הסיומדת dudu –כינוי ממשחק מחשב, כמובן שהיינו יכולים לתת כל שם אחר.

```
draft
                        -השלב במשחק
                                         11
counter - שסופר עוד כמה חיילים נשאר למקם
                                        ____
 משמש בשתי השלבים הראשונים של המשחק
                                        2
                       -מספר השחקנים
                                        player1
 -שם השחקן -,ת.ז, סדר בתור השקחנים ,צבע
                                        1234
  מספר חיילי בונוס, מספר קלפים, במידה ויש
             . קלפים – ישמר סוגי הקלפים
                                        vellow
                                        14
Joker, Solider, Cannon, Horse, Default
                                        player2
                                        5678
                                        1
                                        blue
                                         13
                                        Venezuela
     -הארץ , מספר החיילים הממקומים אליה
                          הבעלים שלה
                                        player2
                                         ____
                                        Brazil
                                        player1
                                         ____
                                        Peru
                                        1
                                        player1
```

<u>סטטיסטקות</u>:

בתוך המשחק אנחנו שומרים סטטיסטקות כלליות על השחקנים –השחקן עם הכי הרבה נצחונות,השחקן עם הכי הרבה הפסדים השחקן ששיחק את מספר המשחקים הרב ביותר והשחקן ששיחק הכי הרבה זמן כמו כן הזמן הממוצע של המשחק של השחקנים . בתחילת כל משחק ניתן ששיחק הכי הרבה זמן כמו כן הזמן הממוצע של המשחק של השחקנים . בתחילת כל משחק או בסטטיסטיקה לבחור אם לצפות בסטטיסטיקה הdefault . במהלך המשחק ישנה אפשרות לשמור לקובץ שנשמרה שלא קשור לסטטיסטיקה הdefault . במהלך המשחק ישנה אפשרות לשמור לקובץ סטטיסטיקה נפרדת שהיא לא ה default – השמירה מתבצעת ע"י לחיצה על לחצן save או ע"י לחיצה על לחצן exit ובמהלך המשחק כל פעם ששחקן מפסיד או מנצח נשמרים נתוניו. כמו כן בנוסף לנתונים כללים ניתן לראות גם נתונים על כל שחקן ששיחק עד אותו הרגע במשחק.

הקובץ ששומר את נתוני המשחק:

| ים | מספר משחק | מן משחק(sec) | הפסדים ז | נצחונות | ic | שם השחקן I |
|----|-----------|--------------|----------|---------|-------|-------------|
| | 30 | 12592 | 0 | 24 | 1234 | player4 |
| | 31 | 13096 | 6 | 12 | 4567 | player3 |
| | 6 | 11944 | 0 | 0 | 2468 | player2 |
| | 7 | 7239 | 0 | 0 | 1357 | player1 |
| | 8 | 1488 | 0 | 0 | 12345 | player12 |
| | 0 | 268 | 0 | 0 | 66 | gal |
| | 1 | 1 | 0 | 0 | 1234 | shoki shalo |

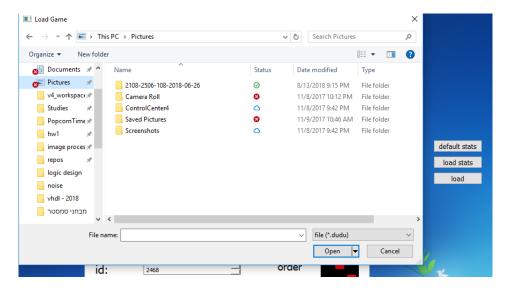


ממשק משתמש:

תפריט



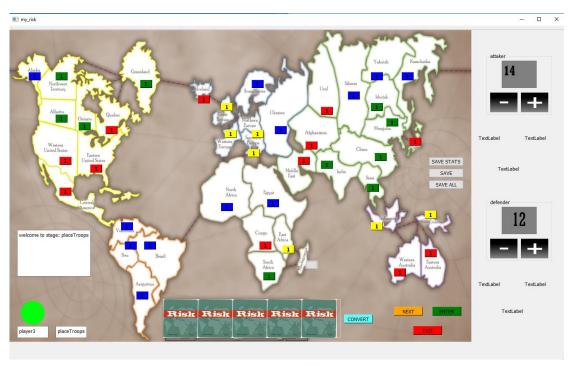
בחירת קובץ



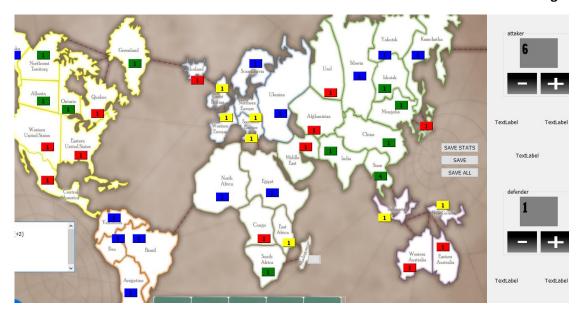
placeMap stage:



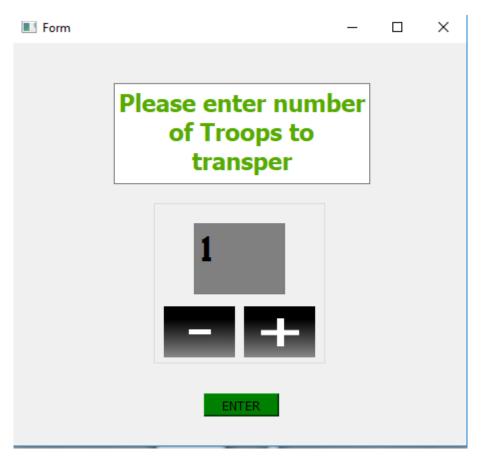
placeTroops stage:



Draft stage:



Move:



delPlayer:



playerWon:



מקורות: <u>מקורות:</u>

- מצגות הקורס תכנות מכוון אובייקטים 2019 31695
- /http://www.cplusplus.com/doc/tutorial/files :Writing of a file
 - :Using static members •

https://www.tutorialspoint.com/cplusplus/cpp_static_members.htm

- https://stackoverflow.com/questions/25201131/writing-csv- :Write to CSV file files-from-c
 - /https://www.geeksforgeeks.org/the-c-standard-template-library-stl :STL •
 - cpp ducmantation https://en.cppreference.com/w/cpp/container/vector
- youtube בנוסף לדוקומנטציה נעזרנו הרבה בסרטונים -qt ducmantation <u>-https://doc.qt.io/</u> המדגימים פעולות שונות בסביבה ובפורם של qt.

main.cpp

C:\risk_final\risk_final\main.cpp

```
1 #include "risk.h"
2 #include "menu.h"
 3 #include <QtWidgets/QApplication>
 4
 5 int main(int argc, char *argv[])
 6 {
 7
        QApplication a(argc, argv);
 8
        menu m;
        m.show();
 9
10
11
       return a.exec();
12 }
13
```

Page 23

C:\risk_final\risk_final\menu.h

```
1 #pragma once
2 #include <QWidget>
3 #include "statsGuiMain.h"
4 #include "risk.h"
5 #include "Cube.h"
 6 #include <iostream>
7 class menu : public QWidget {
       Q_OBJECT
8
9
10 public:
11
       menu(QWidget * parent = Q_NULLPTR);
12
       ~menu();
13 private:
14
       Ui::menu ui;
15
       risk w;
16
      statsGuiMain s;
17
       Cube cubes[4];
18
       Player players[4];
19
      Player* hist[6][4];
20
       int numberOfPlayers;
21
       bool toss;
22
       bool defaultStats_;
23 private slots:
24
       void end();
25
       void spin(int);
26
       void cube();
27
       void LOAD();
28
       void loadStats();
29
       void defalutStats();
30
31 };
32
```

Page 24

C:\risk_final\risk_final\risk_final\menu.cpp

```
#include "menu.h"
 2 #include <time.h>
 3 #include "Cube.h"
 4 #include "windows.h."
 5 #include "qmediaplayer.h"
 6 #include "qmediaplaylist.h"
 7 menu::menu(QWidget * parent) : QWidget(parent), w(4) {
 8
        srand(time(NULL));
 9
        ui.setupUi(this);
        QMediaPlaylist *playlist = new QMediaPlaylist();
10
11
        playlist->addMedia(QUrl("qrc:/new/Resources/Risk.mp3"));
12
        playlist->setPlaybackMode(QMediaPlaylist::Loop);
        OMediaPlayer* music = new OMediaPlayer();
13
        music->setPlaylist(playlist);
14
15
        music->play();
        this->defaultStats = false;
16
        QPixmap image("Resources/risk_menu.jpg");
17
18
        QLabel imageLabel;
19
        imageLabel.setPixmap(image);
20
        imageLabel.show();
21
        this->toss = false;
22
        this->numberOfPlayers = 4;
        QObject::connect(this->ui.FINISH, &QPushButton::clicked, [=] {end(); });
23
24
        QObject::connect(this->ui.cube, &QPushButton::clicked, [=] {cube(); });
25
        QObject::connect(this->ui.LOAD, &QPushButton::clicked, [=] {LOAD(); });
        QObject::connect(this->ui.loadStats, &QPushButton::clicked, [=] {loadStats →
26
          (); });
27
        cubes[0].setLabel(this->ui.cubeP1);
28
        cubes[1].setLabel(this->ui.cubeP2);
29
        cubes[2].setLabel(this->ui.cubeP3);
30
        cubes[3].setLabel(this->ui.cubeP4);
31
        for (int i = 0; i < 4; i++)
32
            cubes[i].hide();
33
        Cube::setPath("Resources/cubes/");
       this->ui.cubeP4->setScaledContents(true);
        //QObject::connect(this->ui.spinBox, SIGNAL(QSpinBox::valueChanged()),
35
          this, SLOT(spin())); -> set in the gui.
36
        this->ui.colorP1->setCurrentIndex(3);
37
        this->ui.colorP2->setCurrentIndex(2);
38
        this->ui.colorP3->setCurrentIndex(1);
39
        this->ui.colorP4->setCurrentIndex(0);
40
        this->s.setStats(this->w.loadStats());
41
42
        //this->defaultStats = true;
43
        //this->loadStats();
        //this->s.setStats(this->w.loadStats());
44
45
        //this->s == nullptr;
46 }
47
48 menu::~menu() {
49
50 }
51 void menu::defalutStats()
52 {
53
        this->s.setFocus();
54
        this->s.show();
```

```
55
56
         //this->defaultStats = true;
57
         //this->loadStats();
58 }
59 void menu::spin(int numberOfPlayers)
60
    {
         this->numberOfPlayers = numberOfPlayers;
61
62
63
         //HIDE
64
         if (numberOfPlayers < 4) {</pre>
65
             this->ui.groupBox4->hide();
 66
         if (numberOfPlayers < 3) {</pre>
67
68
             this->ui.groupBox3->hide();
 69
70
71
         //SHOW
72
         if (numberOfPlayers > 3) {
73
             this->ui.groupBox4->show();
74
         }
75
         if (numberOfPlayers > 2) {
76
             this->ui.groupBox3->show();
 77
         }
78
79
    }
80
81 void menu::cube()
82 {
83
         this->toss = true;
         std::string num2str[] = { "first", "second", "third", "fourth" };
84
85
         for (int i = 0; i < 6; i++)
86
         {
87
             for (int j = 0; j < 4; j++)
88
                 hist[i][j] = nullptr;
89
         }
90
         int iD = std::stoi(this->ui.idP1->toPlainText().toStdString());
91
         players[0] = Player(this->ui.nameP1->toPlainText().toStdString(), this-
           >ui.colorP1->currentText().toStdString(), 0,iD);
92
         players[1] = Player(this->ui.nameP2->toPlainText().toStdString(), this-
                                                                                      P
           >ui.colorP2->currentText().toStdString(), 0,std::stoi(this->ui.idP2-
                                                                                      P
           >toPlainText().toStdString()));
         players[2] = Player(this->ui.nameP3->toPlainText().toStdString(), this-
93
           >ui.colorP3->currentText().toStdString(), 0, std::stoi(this->ui.idP3-
                                                                                      ₽
           >toPlainText().toStdString()));
         players[3] = Player(this->ui.nameP4->toPlainText().toStdString(), this-
94
                                                                                      P
           >ui.colorP4->currentText().toStdString(), 0, std::stoi(this->ui.idP4-
           >toPlainText().toStdString()));
95
         while (!this->w.empty())
96
             this->w.pop();
97
98
         for (int i = 0; i < this->numberOfPlayers; i++) {
99
             this->cubes[i].random();
100
101
         for (int i = 0; i < this->numberOfPlayers; i++)
102
103
             int j = 0;
```

```
C:\risk_final\risk_final\risk_final\menu.cpp
104
             while (hist[this->cubes[i].getValue() - 1][j]) j++;
105
             hist[this->cubes[i].getValue() - 1][j] = &this->players[i];
106
         }
107
         int id = 1;
         for (int i = 5; i >= 0; i--)
108
109
110
             int len = 0;
             while (hist[i][len]) len++;
111
112
             if (len == 0) continue;
113
             while (len)
114
                 hist[i][len-1]->SetOrder(id);
115
                 this->w.push(*hist[i][len-1]);
116
117
                 id++;
118
                 len--;
119
             }
120
         }
121
122
         this->ui.pic1_ord->setText(num2str[this->players[0].GetOrder()-1].c_str
           ());
123
         this->ui.pic2 ord->setText(num2str[this->players[1].GetOrder() - 1].c str >
           ());
124
         if (this->numberOfPlayers > 2) {
125
             this->ui.pic3_ord->setText(num2str[this->players[2].GetOrder() -
               1].c_str());
126
127
         if (this->numberOfPlayers > 3) {
128
             this->ui.pic4_ord->setText(num2str[this->players[3].GetOrder() -
               1].c_str());
129
         }
130 }
131
132 void menu::LOAD()
133 {
134
         if (!this->w.LOAD())
135
             return;
136
         this->close();
137
         this->w.setFocus();
138
         this->w.show();
139
     }
140
141 void menu::loadStats()
142 {
143
         //if(this->s != nullptr)
144
         // this->s.clear();
145
         if (this->defaultStats_ == true) {
146
             this->s.setStats(this->w.loadStats());
147
             this->s.setFocus();
148
             this->s.show();
149
             return;
150
151
         QString fileName = QFileDialog::getOpenFileName(this,
152
             tr("Load Game"),
153
             tr("file (*.csv);;All Files (*)"));
154
         if (fileName == "")
155
```

```
C:\risk_final\risk_final\menu.cpp
156
            return;
157
        this->s.setStats(this->w.loadStats(fileName.toStdString()));
158
        this->s.setFocus();
159
        this->s.show();
160 }
161
162
163 void menu::end() {
164
        if (this->toss == false) {
165
            this->ui.info->setText("I see you got NA education .. \nplease toss
              the cubes!");
166
            return;
167
        }
        this->w.set_number_of_player(this->numberOfPlayers);
168
        this->close();
169
        this->w.showInfo();
170
171
        this->w.initTroops();
        this->w.showTroops();
172
173
        this->w.set_startTime(std::time(0));
174
        this->w.updateStage();
175
        this->w.setFocus();
176
        this->w.show();
177 }
```

```
1 #pragma once
 2 #include "Player.h"
 3 #include "Country.h"
 4 #define num_of_conutries 42
 5 #define TOTAL_TROOPS 100
 6 #define TROOP4COUNTRY 3
 7 #include <QtWidgets/QMainWindow>
 8 #include <QMediaPlayer>
 9 #include "ui_my_risk.h"
10 #include "Qdebug.h"
11 #include <Qsignalmapper.h>
12 #include <utility>
13 #include <queue>
14 #include <fstream>
15 #include <qfiledialog.h>
16 #include <functional>
17 #include <iostream>
18 #include "Cube.h"
19 #include "Move.h"
20 #include "Stage.h"
21 #include "Attack.h"
22 #include "Draft.h"
23 #include "Reinforcement.h"
24 #include "PlaceMap.h"
25 #include "PlaceTroops.h"
26 #include "PlayerStats.h"
27 #include <chrono>
28 #include <ctime>
29 #include "Mess.h"
30 #define _maxCardsPerPlayer_ 5
31 class risk :public QMainWindow {
32
       Q OBJECT
33 public:
34
       risk(int numberOfPlayers, QWidget *parent = 0);
       char get_number_of_player();
35
       void set_number_of_player(char num);
36
37
       void incPlayer();
38
       void initTroops();
39
       void showTroops();
40
41
       bool empty() const;
42
       void pop();
43
       void push(const Player&);
44
45
       void save(std::ofstream& file);
46
       void load(std::ifstream& file);
47
       PlayerStats* loadStats(std::string fileName = "Resources/stats.csv");
48
       void loadStatsGui();
49
       void showInfo();
50
       void updateStage();
51
       time_t get_startTime() const;
52
       void set startTime(const time t);
53 private:
54
       std::queue<Player> queue;
55
       Land SouthAmerica, NorthAmerica, Europe, Asia, Australia, Africa;
       std::vector<QPushButton*> SA buttons;
```

```
C:\risk_final\risk_final\risk.h
       std::vector<QPushButton*> NA_buttons;
58
       std::vector<QPushButton*> EU_buttons;
59
       std::vector<QPushButton*> AS_buttons;
60
       std::vector<QPushButton*> AU_buttons;
61
       std::vector<QPushButton*> AF_buttons;
62
63
       char numberOfPlayers;
       int numberOfCountries; //set in ctor and does not change
64
65
       int troopsPerPlayer;
66
       int counter;
                       //counter in placeMap and placeTroops()
67
       bool newCard;
       std::vector<Card> cards;
68
69
       void cardClick(int sel = 0);
70
71
       std::string stage;
72
       Stage* currentStage;
73
       std::map<std::string, std::string> stage2preStage;
74
       PlayerStats stats;
75
       time_t startTime;
76
       void showMess(std::string);
77
       Mess mess;
78
       std::map<std::string, Player> playersMap;
79
       Move moveUi;
80
       void saveStatsIn(std::string = "Resources/stats.csv",bool all = 0);
81 public slots:
82
       bool LOAD();
83
       void SAVE();
84
       void ENTER();
85
       void SAVE_ALL();
86
       void saveStats();
87
       void EXIT();
88
      void nextState();
89
90 private:
91
       Ui::my_riskClass ui;
92 };
```

```
C:\risk_final\risk_final\risk.cpp
 1 #include "risk.h"
 3 risk::risk(int numberOfPlayers, QWidget *parent) :troopsPerPlayer
                                                                                  P
      (TOTAL_TROOPS / numberOfPlayers), QMainWindow(parent), cards
      { Card::cardType::Default, Card::cardType::Default, Card::cardType::Default, →
       Card::cardType::Default,Card::cardType::Default }
 4
   {
 5
        ui.setupUi(this);
        //SouthAmerica, NorthAmerica, Europe, Asia, Australia, Africa;
 6
 7
        Country::setUI(&this->ui);
 8
        newCard = false;
 9
        this->numberOfPlayers = numberOfPlayers;
10
        this->counter = 0;
        this->stage = "placeMap";
11
        QObject::connect(this->ui.ENTER, &QPushButton::clicked, [=] {ENTER(); });
12
13
        QObject::connect(this->ui.STATS, &QPushButton::clicked, [=] {saveStats
          ();});
14
15
        //this->currentStage->setNextEn(true);
        stage2preStage["placeMap"] = "initial";
16
        stage2preStage["placeTroops"] = "placeMap";
17
        stage2preStage["draft"] = "placeTroops";
18
        stage2preStage["attack"] = "draft";
19
20
        stage2preStage["reinforcement"] = "attack";
        stage2preStage["default"] = "reinforcement";
21
22
23
24
        this->cards[0].setButton(this->ui.card1);
25
        this->cards[1].setButton(this->ui.card2);
26
        this->cards[2].setButton(this->ui.card3);
27
        this->cards[3].setButton(this->ui.card4);
28
        this->cards[4].setButton(this->ui.card5);
29
        this->ui.cubeP 1->hide();
30
        this->ui.cubeP 2->hide();
31
        this->ui.cubeP 3->hide();
32
        this->ui.cubeP2_1->hide();
        this->ui.cubeP2_2->hide();
33
34
        this->ui.cubeP2 3->hide();
        this->ui.TROOPS_2->setVisible(false);
35
36
37
        SA
                                                 std::vector< std::string> SA{ "Venezuela", "Brazil", "Peru", "Argentina" };
39
        this->SouthAmerica = Land(SA, { 2,"WellDone you conquered SouthAmerica
          (+2)" });
40
        this->SA buttons.reserve(SA.size());
41
42
        this->SA_buttons = {
43
            this->ui.south_america_1 ,this->ui.south_america_2 ,this-
              >ui.south america 3,
44
            this->ui.south_america_4
45
        };
46
        for (int i = 0; i < this->SA buttons.size(); ++i) {
47
48
            SouthAmerica.countries[i].setButton(this->SA_buttons[i]);
49
            QObject::connect(this->SA_buttons[i], &QPushButton::clicked, [=]
              {SouthAmerica.countries[i].init(); });
```

```
C:\risk_final\risk_final\risk.cpp
50
51
52
        NA
                                              std::vector< std::string> NA{ "Alaska","Northwest
53
         Territory", "Greenland", "Alberta", "ontario", "Quebec", "Western United
                                                                             P
         States","Eastern United States","Central America" };
        this->NorthAmerica = Land(NA, { 5 ,"WellDone you conquered NorthAmerica
54
          (+5) - about 7.5% of the world's population\n"});
55
56
        this->NA_buttons.reserve(NA.size());
57
        this->NA buttons = {
58
           this->ui.north america 1 ,this->ui.north america 2 ,this-
             >ui.north america 3,
59
           this->ui.north_america_4 ,this->ui.north_america_5 ,this-
             >ui.north america 6,
           this->ui.north_america_7 ,this->ui.north_america_8 ,this-
60
             >ui.north america 9
61
        };
62
63
        for (int i = 0; i < this->NA buttons.size(); ++i) {
64
           NorthAmerica.countries[i].setButton(this->NA buttons[i]);
           QObject::connect(this->NA_buttons[i], &QPushButton::clicked, [=]
65
             {NorthAmerica.countries[i].init(); });
66
        }
67
        68
                                        ΑF
69
        std::vector< std::string> AF{ "North Africa" ,"Egypt" ,"Cango" ,"East
         Africa", "South Africa", "Madagascar" };
70
        this->Africa = Land(AF,{3, "WellDone you conquered Africa(+3)"});
71
72
        this->AF buttons.reserve(AF.size());
73
        this->AF buttons = {
74
           this->ui.africa ,this->ui.africa_2 ,this->ui.africa_3,
75
           this->ui.africa 4 ,this->ui.africa 5 ,this->ui.africa 6
76
        };
77
        for (int i = 0; i < this->AF_buttons.size(); ++i) {
78
79
           Africa.countries[i].setButton(this->AF_buttons[i]);
80
           QObject::connect(this->AF_buttons[i], &QPushButton::clicked, [=]
             {Africa.countries[i].init(); });
81
        }
82
83
        EU
        std::vector< std::string> EU{ "Iceland", "Great
84
         Berlin", "Scandinavia", "Western Europe", "Soutern Europe", "Northern
         Europe" ,"Ukraine" };
        this->Europe = Land(EU, { 5, "WellDone you conquered Europe (+5), did you
85
         know 28 of the European states belong to the European Union?" });
86
87
        this->EU_buttons.reserve(EU.size());
88
        this->EU_buttons = {
89
           this->ui.europe 1 ,this->ui.europe 2 ,this->ui.europe 3,
           this->ui.europe_4 ,this->ui.europe_5 ,this->ui.europe_6,this-
90
             >ui.europe 7
91
        };
92
```

```
C:\risk_final\risk_final\risk.cpp
 93
        for (int i = 0; i < this->EU buttons.size(); ++i) {
 94
            Europe.countries[i].setButton(this->EU buttons[i]);
 95
            QObject::connect(this->EU_buttons[i], &QPushButton::clicked, [=]
                                                                                P
              {Europe.countries[i].init(); });
        }
 96
 97
 98
        AS
                                                99
        std::vector< std::string> AS
          { "Kamchatka", "Yakutsk", "Siberia", "Irkutsk", "Mongolia", "China", "India >
           ,"Siam","Ural","Afghanistan","Middle East","Japan" };
        this->Asia = Land(AS,{7, "WellDone you conquered Asia (+5)(+7)"});
100
101
        this->AS buttons.reserve(AS.size());
102
103
        this->AS_buttons = {
            this->ui.asia ,this->ui.asia_2,this->ui.asia_3,this->ui.asia_4,this-
104
              >ui.asia 5,
105
            this->ui.asia_6 ,this->ui.asia_7 ,this->ui.asia_8,this-
              >ui.asia_9,this->ui.asia_10,this->ui.asia_11, this->ui.asia_12
106
        };
107
108
        for (int i = 0; i < this->AS buttons.size(); ++i) {
109
            Asia.countries[i].setButton(this->AS buttons[i]);
            QObject::connect(this->AS buttons[i], &QPushButton::clicked, [=]
110
              {Asia.countries[i].init(); });
        }
111
        112
                                          ΑU
        std::vector< std::string> AU{ "Western Australia" ,"Eastern
113
          Australia" ,"New Gulinea" ,"Indonesia" };
114
        this->Australia = Land(AU, { 2,"WellDone you conquered Australia (+2)" });
115
116
        this->AU buttons.reserve(AU.size());
        this->AU_buttons = {
117
118
            this->ui.australia ,this->ui.australia_2,
119
            this->ui.australia 3,this->ui.australia 4
120
        };
        for (int i = 0; i < this->AU_buttons.size(); ++i) {
121
            Australia.countries[i].setButton(this->AU_buttons[i]);
122
123
            QObject::connect(this->AU_buttons[i], &QPushButton::clicked, [=]
              {Australia.countries[i].init(); });
124
        this->numberOfCountries = AU.size() + AS.size() + NA.size() + SA.size() + >
125
          AF.size() + EU.size();
126
        //set neihbors
        /*
127
        */
128
129
        // in order to orginaze the connection easily we map the country names
          temprarly to pointer to their objects.
130
131
        std::map<std::string, Country*> m;
132
        m["Alaska"] = &(this->NorthAmerica.countries[0]);
        m["Alberta"] = &(this->NorthAmerica.countries[3]);
133
        m["Central America"] = &(this->NorthAmerica.countries[8]);
134
        m["Eastern United States"] = &(this->NorthAmerica.countries[7]);
135
136
        m["Greenland"] = &(this->NorthAmerica.countries[2]);
        m["Northwest Territory"] = &(this->NorthAmerica.countries[1]);
137
138
        m["Ontario"] = &(this->NorthAmerica.countries[4]);
```

```
C:\risk_final\risk_final\risk.cpp
139
         m["Quebec"] = &(this->NorthAmerica.countries[5]);
140
         m["Western United States"] = &(this->NorthAmerica.countries[6]);
141
142
         m["Argentina"] = &(this->SouthAmerica.countries[3]);
143
144
         m["Brazil"] = &(this->SouthAmerica.countries[1]);
145
         m["Peru"] = &(this->SouthAmerica.countries[2]);
         m["Venezuela"] = &(this->SouthAmerica.countries[0]);
146
147
         m["Great Britain"] = &(this->Europe.countries[1]);
148
149
         m["Iceland"] = &(this->Europe.countries[0]);
         m["Northern Europe"] = &(this->Europe.countries[5]);
150
         m["Scandinavia"] = &(this->Europe.countries[2]);
151
152
         m["Southern Europe"] = &(this->Europe.countries[4]);
         m["Ukraine"] = &(this->Europe.countries[6]);
153
154
         m["Western Europe"] = &(this->Europe.countries[3]);
155
         m["Cango"] = &(this->Africa.countries[2]);
156
157
         m["East Africa"] = &(this->Africa.countries[3]);
158
         m["Egypt"] = &(this->Africa.countries[1]);
159
         m["Madagascar"] = &(this->Africa.countries[5]);
         m["North Africa"] = &(this->Africa.countries[0]);
160
         m["South Africa"] = &(this->Africa.countries[4]);
161
         //std::vector< std::string> AS{ "Kamchatka0", "Yakutsk1", "Siberia2",
   "Irkutsk3", "Mongolia4", "China5", "India6", "Siam7", "Ural8",
162
           "Afghanistan9", "Middle East10", japen[11] }; Africa
163
         m["Afghanistan"] = &(this->Asia.countries[9]);
164
         m["China"] = &(this->Asia.countries[5]);
165
         m["India"] = &(this->Asia.countries[6]);
           "Irkutsk"] = &(this->Asia.countries[3]);
166
167
         m["Japan"] = &(this->Asia.countries[11]);
168
         m["Kamchatka"] = &(this->Asia.countries[0]);
         m["Middle East"] = &(this->Asia.countries[10]);
169
170
         m["Mongolia"] = &(this->Asia.countries[4]);
171
         m["Siam"] = &(this->Asia.countries[7]);
172
         m["Siberia"] = &(this->Asia.countries[2]);
         m["Ural"] = &(this->Asia.countries[8]);
173
174
         m["Yakutsk"] = &(this->Asia.countries[1]);
175
176
         //std::vector< std::string> AU{ "Western Australia", "Eastern Australia", ▶
           "New Gulinea", "Indonesia" }; Australia
         m["Eastern Australia"] = &(this->Australia.countries[0]);
177
178
         m["Indonesia"] = &(this->Australia.countries[3]);
         m["New Guinea"] = &(this->Australia.countries[2]);
179
180
         m["Western Australia"] = &(this->Australia.countries[1]);
181
182
183
         //SA
184
         m["Argentina"]->setNehibors({ m["Brazil"], m["Peru"] });
185
         m["Brazil"]->setNehibors({ m["Peru"], m["Venezuela"], m["Argentina"], m
           ["North Africa"] });
         m["Peru"]->setNehibors({ m["Venezuela"], m["Brazil"] ,m["Argentina"] });
186
         m["Venezuela"]->setNehibors({ m["Central America"], m["Peru"], m
187
           ["Brazil"] });
188
         //eu
         m["Great Britain"]->setNehibors({ m["Iceland"], m["Scandinavia"], m
189
```

```
C:\risk_final\risk_final\risk.cpp
           ["Western Europe"], m["Northern Europe"] });
        m["Iceland"]->setNehibors({ m["Greenland"], m["Great Britain"] });
190
191
        m["Northern Europe"]->setNehibors({ m["Ukraine"], m["Soutern Europe"],m
                                                                                    P
           ["Western Europe"],m["Great Britain"] });
192
        m["Scandinavia"]->setNehibors({ m["Ukraine"], m["Great Britain"] });
193
        m["Southern Europe"]->setNehibors({ m["North Africa"], m["Egypt"] ,m
           ["Ukraine"] ,m["Western Europe"],m["Northern Europe"] });
        m["Ukraine"]->setNehibors({ m["Scandinavia"], m["Northern Europe"] ,m
194
           ["Southern Europe"] ,m["Middle East"], m["Ural"],m["Afghanistan"] });
195
        m["Western Europe"]->setNehibors({ m["North Africa"], m["Southern
                                                                                    P
           Europe"] , m["Northern Europe"] , m["Great Britain"] });
196
        m["North Africa"]->setNehibors({ m["Brazil"], m["Cango"],m["Egypt"], m
197
           ["Southern Europe"], m["Western Europe"] });
        m["Egypt"]->setNehibors({ m["Middle East"], m["Southern Europe"], m["North →
198
            Africa"], m["Cango"],m["East Africa"] });
199
        m["Cango"]->setNehibors({ m["South Africa"], m["East Africa"] ,m
           ["Egypt"] ,m["North Africa"] });
200
        m["East Africa"]->setNehibors({ m["Cango"], m["Egypt"], m["South
           Africa"],m["Madagascar"] });
        m["South Africa"]->setNehibors({ m["Madagascar"], m["East Africa"] ,m
201
           ["Cango"] });
        m["Madagascar"]->setNehibors({ m["East Africa"], m["South Africa"] });
202
203
        //as
204
        m["Afghanistan"]->setNehibors({ m["Ukraine"], m["Middle East"] , m
           ["India"] , m["China"] , m["Ural"] , m["Siberia"] });
        m["China"]->setNehibors({ m["India"], m["Siam"], m["Afghanistan"] , m
205
           ["Mongolia"] , m["Siberia"] , m["Ural"] });
206
        m["India"]->setNehibors({ m["Middle East"], m["Afghanistan"], m["China"],
           m["Siam"] });
        m["Irkutsk"]->setNehibors({ m["Mongolia"], m["Siberia"] , m["Yakutsk"] , m >
207
           ["Kamchatka"] });
        m["Japan"]->setNehibors({ m["Mongolia"], m["Kamchatka"] });
208
209
        m["Kamchatka"]->setNehibors({ m["Irkutsk"], m["Yakutsk"] , m["Mongolia"],
           m["Japan"],m["Alaska"] });
        m["Middle East"]->setNehibors({ m["India"], m["Afghanistan"] , m
210
           ["Ukraine"], m["Egypt"] });
211
        m["Mongolia"]->setNehibors({ m["Japan"], m["China"] , m["Irkutsk"] , m
           ["Siberia"] });
        m["Siam"]->setNehibors({ m["China"], m["India"] , m["Indonesia"] });
212
        m["Siberia"]->setNehibors({ m["Yakutsk"], m["Irkutsk"], m["Mongolia"], m
213
           ["China"] , m["Afghanistan"] , m["Ural"] });
        m["Ural"]->setNehibors({ m["Siberia"], m["Ukraine"] , m["Afghanistan"] });
214
        m["Yakutsk"]->setNehibors({ m["Kamchatka"], m["Irkutsk"], m["Siberia"] });
215
216
217
        m["Eastern Australia"]->setNehibors({ m["Western Australia"], m["New
           Guinea"] });
218
        m["Indonesia"]->setNehibors({ m["Siam"], m["New Guinea"], m["Western
           Australia"] });
219
        m["New Guinea"]->setNehibors({ m["Indonesia"], m["Eastern Australia"], m
           ["Western Australia"] });
        m["Western Australia"]->setNehibors({ m["New Guinea"], m["Indonesia"], m
220
           ["Eastern Australia"] });
221
222
        //na
223
        m["Alaska"]->setNehibors({ m["Northwest Territory"], m["Kamchatka"] });
```

```
C:\risk_final\risk_final\risk.cpp
224
         m["Alberta"]->setNehibors({ m["Northwest Territory"], m["Ontario"], m
           ["Western United States"], m["Alaska"] });
         m["Central America"]->setNehibors({ m["Venezuela"], m["Eastern United
225
                                                                                    P
           States"], m["Western United States"] });
         m["Eastern United States"]->setNehibors({ m["Quebec"], m["Ontario"], m
226
           ["Western United States"], m["Central America"] });
227
         m["Greenland"]->setNehibors({ m["Northwest Territory"], m["Ontario"], m
                                                                                    P
           ["Quebec"], m["Iceland"] });
228
         m["Northwest Territory"]->setNehibors({ m["Alaska"], m["Alberta"], m
           ["Ontario"] });
         m["Ontario"]->setNehibors({ m["Northwest Territory"], m["Alberta"], m
229
           ["Western United State"], m["Eastern United States"] , m["Quebec"] });
         m["Quebec"]->setNehibors({ m["Ontario"], m["Greenland"], m["Eastern United ₹
230
            States"] });
         m["Western United States"]->setNehibors({ m["Alberta"], m["Eastern United →
231
           States"], m["Ontario"], m["Central America"] });
232
233
234
235
         this->queue.push(Player());
236
         this->currentStage = new PlaceMap(this->ui, this->queue.front(), this-
           >moveUi, this->counter);
237
         ui.textEdit->setText("welocme to the game");
238 }
239
240
241 void risk::incPlayer()
242 {
243
         this->queue.push(this->queue.front());
244
         this->queue.pop();
245
         QString picPath = QString("Resources/players/") + QString(this-
           >queue.front().getColor().c_str()) + ".png";
246
         QPixmap pic(picPath);//("Resources/cubes/1.png");
247
         this->ui.colorP->setPixmap(pic);
248
         this->ui.colorP->show();
249
    }
250
251 void risk::cardClick(int sel)
252 {
253
         this->cards[sel].click();
254 }
255
256 void risk::showMess(std::string txt)
257
258
         this->mess.setMess(txt);
259
         this->mess.setFocus();
260
         this->mess.show();
261 }
262
263 void risk::showTroops()
264 {
265
         int bonus = this->queue.front().getBonusTroops();
266
         QString text = std::to string(bonus).c str();
267
         this->ui.troopsP->setText(text);
268 }
269
```

```
270 bool risk::empty() const
271 {
272
         return this->queue.empty();
273 }
274
275 void risk::pop()
276 {
277
         this->queue.pop();
278 }
279
280 void risk::push(const Player & player)
281 {
         this->queue.push(player);
282
283 }
284
285 void risk::initTroops()
286 {
287
         //y=-5x+50;
                         -> this is the real but it is TOO MUCH!
288
         int troops = -5 * this->numberOfPlayers + 35;
         for (int i = 0; i < this->numberOfPlayers; i++) {
289
290
             this->queue.front().setBonusTroops(troops);
291
             incPlayer();
292
         }
293
    }
294
295 void risk::nextState()
296 {
297
         if (this->currentStage->getNexEn() == false)
298
             return;
299
         this->currentStage->afterAction();
300
         if (this->currentStage->getNexEn() == false)//case draft that you have
           cards to convert
301
             return;
302
         delete this->currentStage;
         if (this->stage == "initial") {
303
             this->stage = "placeMap";
304
             this->currentStage = new PlaceMap(this->ui, this->queue.front(), this- →
305
               >moveUi, this->counter);
306
307
         else if (this->stage == "placeMap") {
308
             this->stage = "placeTroops";
             this->currentStage = new PlaceTroops(this->ui, this->queue.front(),
309
               this->moveUi, this->counter);
310
311
         else if (this->stage == "placeTroops") {
             this->stage = "draft";
312
             this->currentStage = new Draft(this->ui, this->queue.front(), this-
313
               >moveUi, this->cards,_maxCardsPerPlayer_);
314
             std::vector<Land*> lands = { &this->SouthAmerica,
315
                                         &this->NorthAmerica,
316
                                         &this->Asia,
317
                                         &this->Australia,
318
                                         &this->Africa,
319
                                         &this->Europe,
320
             };
321
             dynamic cast<Draft*>(this->currentStage)->setLands(lands);
```

```
C:\risk_final\risk_final\risk.cpp
322
323
             this->ui.PLUS->disconnect();
324
             this->ui.MINUS->disconnect();
325
             QObject::connect(this->ui.PLUS, &QPushButton::clicked, [=]
               {currentStage->plus(); });
             QObject::connect(this->ui.MINUS, &QPushButton::clicked, [=]
326
               {currentStage->minus(); });
327
328
             this->ui.card1->disconnect();
329
             this->ui.card2->disconnect();
330
             this->ui.card3->disconnect();
331
             this->ui.card4->disconnect();
332
             this->ui.card5->disconnect();
333
             QObject::connect(this->ui.card1, &QPushButton::clicked, [=] {cards
334
               [0].click(); });
335
             QObject::connect(this->ui.card2, &QPushButton::clicked, [=] {cards
               [1].click(); });
336
             QObject::connect(this->ui.card3, &QPushButton::clicked, [=] {cards
               [2].click(); });
             QObject::connect(this->ui.card4, &QPushButton::clicked, [=] {cards
337
                                                                                     P
               [3].click(); });
338
             QObject::connect(this->ui.card5, &QPushButton::clicked, [=] {cards
               [4].click(); });
339
             this->ui.CONVERT->disconnect();
340
             QObject::connect(this->ui.CONVERT, &QPushButton::clicked, [=]
341
               {dynamic_cast<Draft*>(currentStage)->convert(); });
342
         else if (this->stage == "draft") {
343
344
             this->stage = "attack";
345
             this->ui.CONVERT->disconnect();
346
             int idx = 0;
347
             while (this->cards[idx++] != Card(Card::Default));
             idx--;
348
             this->currentStage = new Attack(this->ui, this->queue.front(), this-
349
               >moveUi, &(this->cards[idx]));
350
             this->ui.PLUS->disconnect();
351
             this->ui.PLUS_2->disconnect();
352
             this->ui.MINUS->disconnect();
353
             this->ui.MINUS 2->disconnect();
354
             QObject::connect(this->ui.PLUS, &QPushButton::clicked, [=]
355
               {currentStage->plus(); });
356
             QObject::connect(this->ui.MINUS, &QPushButton::clicked, [=]
               {currentStage->minus(); });
             QObject::connect(this->ui.PLUS_2, &QPushButton::clicked, [=]
357
               {currentStage->plus(1); });
358
             QObject::connect(this->ui.MINUS_2, &QPushButton::clicked, [=]
               {currentStage->minus(1); });
359
360
         else if (this->stage == "attack") {
361
362
             this->currentStage = new Reinforcement(this->ui, this->queue.front(), →
               this->moveUi);
             this->ui.PLUS->disconnect();
363
```

```
C:\risk_final\risk_final\risk.cpp
364
             this->ui.PLUS 2->disconnect();
365
             this->ui.MINUS->disconnect();
366
             this->ui.MINUS_2->disconnect();
367
             QObject::connect(this->ui.PLUS, &QPushButton::clicked, [=]
368
               {currentStage->plus(); });
             QObject::connect(this->ui.MINUS, &QPushButton::clicked, [=]
369
               {currentStage->minus(); });
             QObject::connect(this->ui.PLUS_2, &QPushButton::clicked, [=]
370
               {currentStage->plus(1); });
             QObject::connect(this->ui.MINUS_2, &QPushButton::clicked, [=]
371
               {currentStage->minus(1); });
372
373
             this->stage = "reinforcement";
374
375
         else if (this->stage == "reinforcement") {
376
             this->stage = "draft";
377
             this->incPlayer();
378
             this->currentStage = new Draft(this->ui, this->queue.front(), this-
               >moveUi, this->cards, _maxCardsPerPlayer_);
379
             std::vector<Land*> lands = { &this->SouthAmerica,
380
                                          &this->NorthAmerica,
381
                                          &this->Asia,
382
                                          &this->Australia,
383
                                          &this->Africa,
384
                                          &this->Europe,
385
             };
386
             dynamic_cast<Draft*>(this->currentStage)->setLands(lands);
387
             Land::landInfo res = dynamic_cast<Draft*>(this->currentStage)-
388
               >calcTroops();
389
             if (res.bonus == -1) { //player need to be delete.
390
                 this->numberOfPlayers--;
391
                 if (this->numberOfPlayers == 1) {
392
                     this->queue.pop();
                     this->showMess(this->queue.front().getName() + " won!");
393
394
                     this->saveStatsIn();
395
                 }
                 else {
396
397
                     this->showMess(this->queue.front().getName() + " kicked out!
                       losser..");
398
                     this->saveStatsIn();
399
                     //this->saveStats();
400
                     this->queue.pop();
401
                     return this->updateStage();
402
                 }
403
             }
404
             this->ui.PLUS->disconnect();
405
             this->ui.PLUS 2->disconnect();
             this->ui.MINUS->disconnect();
406
             this->ui.MINUS_2->disconnect();
407
408
             QObject::connect(this->ui.PLUS, &QPushButton::clicked, [=]
409
               {currentStage->plus(); });
             QObject::connect(this->ui.MINUS, &QPushButton::clicked, [=]
410
                                                                                      P
               {currentStage->minus(); });
```

```
C:\risk_final\risk_final\risk.cpp
411
             OObject::connect(this->ui.PLUS 2, &QPushButton::clicked, [=]
               {currentStage->plus(1); });
             QObject::connect(this->ui.MINUS_2, &QPushButton::clicked, [=]
412
                                                                                      P
               {currentStage->minus(1); });
413
414
415
             this->ui.card1->disconnect();
416
             this->ui.card2->disconnect();
417
             this->ui.card3->disconnect();
418
             this->ui.card4->disconnect();
             this->ui.card5->disconnect();
419
420
             QObject::connect(this->ui.card1, &QPushButton::clicked, [=] {cards
421
               [0].click(); });
             QObject::connect(this->ui.card2, &QPushButton::clicked, [=] {cards
422
               [1].click(); });
423
             QObject::connect(this->ui.card3, &QPushButton::clicked, [=] {cards
               [2].click(); });
424
             QObject::connect(this->ui.card4, &QPushButton::clicked, [=] {cards
               [3].click(); });
425
             QObject::connect(this->ui.card5, &QPushButton::clicked, [=] {cards
                                                                                     P
               [4].click(); });
426
             this->ui.CONVERT->disconnect();
427
428
             QObject::connect(this->ui.CONVERT, &QPushButton::clicked, [=]
               {dynamic_cast<Draft*>(currentStage)->convert(); });
429
         }
430
         this->currentStage->preAction();
431
         this->showInfo();
         this->ui.playerText->setText(this->queue.front().getName().c_str());
432
433
         this->ui.stageText->setText(this->stage.c str());
434 }
435
436
    void risk::save(std::ofstream& file) {
437
         file << this->stage << "\n";
438
         file << this->counter << "\n";</pre>
         file << "----\n";
439
440
         file << std::to_string(this->numberOfPlayers) << "\n";</pre>
441
         std::queue<Player> queue = this->queue;
442
         for (int i = 0; i < this->numberOfPlayers; i++) {
443
             queue.front().save(file);
444
             queue.push(this->queue.front());
445
             queue.pop();
446
         }
447
         this->SouthAmerica.save(file);
448
         this->NorthAmerica.save(file);
449
         this->Europe.save(file);
450
         this->Asia.save(file);
451
         this->Australia.save(file);
452
         this->Africa.save(file);
453 }
454
455 void risk::load(std::ifstream & file)
456 {
457
         std::string tmp;
458
         //load stage
```

511 {
512

std::ifstream file;

```
C:\risk_final\risk_final\risk.cpp
                                                                                   12
513
         file.open(fileName);
514
         this->stats.loadStats(file);
515
         file.close();
516
         return &this->stats;
517 }
518
519 void risk::loadStatsGui()
520 {
521
         QString fileName = QFileDialog::getOpenFileName(this,
522
         tr("Load Game"),
523
            "",
            tr("file (*.csv);;All Files (*)"));
524
         this->loadStats(fileName.toStdString());
525
526 }
527
528 void risk::showInfo()
529 {
530
         QString picPath = QString("Resources/players/") + QString(this-
           >queue.front().getColor().c_str()) + ".png";
531
         QPixmap pic(picPath);//("Resources/cubes/1.png");
532
         this->ui.colorP->setPixmap(pic);
533
         this->ui.colorP->show();
534
    }
535
536 void risk::updateStage()
537 {
538
         this->stage = this->stage2preStage[this->stage];
539
         this->currentStage->setNextEn(true);
540
         this->nextState();
541 }
542
543 time_t risk::get_startTime() const
544 {
545
         return this->startTime;
546 }
547
548 void risk::set_startTime(const time_t time )
549 {
550
         this->startTime = time;
551 }
552
553 void risk::SAVE ALL()
554 {
555
         this->SAVE();
556
         this->saveStats();
557 }
558 ///this function is called in some situation -first it called each time player →
       lose or win.
559 // second it called when player clicked exit butten.
560 // it also called whether the path for save it the default and when it isn't- 🤝
      mean the player chose other path
561 // all variable is flag that send in order to know if the function is called
      with exit
562 // then no one lose or win or when player is kicked out or win during the
                                                                                    P
       game.
563 //all is 0 by default and its called with 1 only when clicked exit and its
```

```
only 1 player-means win.
564 void risk::saveStatsIn(std::string fileName, bool all)
565 {
566
         std::ofstream file;
567
         file.open(fileName);
568
         std::queue<Player> queue = this->queue;
569
         time_t end = std::time(0);
570
         int elapsed_seconds = end - this->startTime;
571
         this->startTime = end;
572
         int len = all ? this->numberOfPlayers : 1;
573
        int win = 0;
                         //nutral
574
         if (this->numberOfPlayers == 1)
575
             win = 1;
                         //win
576
         else if (all == false)
577
             win = -1; //loose
578
         for (int i = 0; i < len; ++i) {
579
             PlayerStatWrap tmp(queue.front(), win, elapsed_seconds);//
                                                                                      P
               player, win, time
580
             this->stats.addToStats(tmp);
581
582
             queue.pop();
         }
583
584
         this->stats.saveStats(file);
585
         file.close();
586
    }
587
588 void risk::SAVE()
589 {
590
         QString fileName = QFileDialog::getSaveFileName(this,
591
             tr("Save Game"),
592
             "",
593
             tr("file (*.dudu);;All Files (*)"));
594
595
         std::ofstream file;
         file.open(fileName.toStdString());
596
597
         this->save(file);
         file.close();
598
599
600 }
601
602 void risk::ENTER()
603 {
604
         if (Country::chosen == nullptr)
605
             this->ui.textEdit->setText(QString("please start the game .."));
606
             return;
607
608
609
         this->currentStage->action();
610
         if (this->currentStage->getEnterEn() == true && (this->stage == "placeMap" →
            || this->stage == "placeTroops")) {
611
             this->incPlayer();
612
             this->updateStage();
             if (this->stage == "placeMap") {
613
614
                 if (this->counter == this->numberOfCountries) {
615
                     this->currentStage->setNextEn(true);
616
                     this->nextState(); //to placeTroops
```

```
C:\risk_final\risk_final\risk.cpp
                                                                                    14
617
618
             }
             else {
619
620
                 if (this->counter == (-5 * this->numberOfPlayers + 35)*this-
                   >numberOfPlayers) {
621
                     this->currentStage->setNextEn(true);
                     this->nextState(); //to draft
622
623
                 }
             }
624
625
         }
626 }
627
628
629 bool risk::LOAD()
630 {
         QString fileName = QFileDialog::getOpenFileName(this,
631
             tr("Load Game"),
632
             "",
633
             tr("file (*.dudu);;All Files (*)"));
634
635
         if (fileName == "")
636
             return 0;
         std::ifstream file;
637
        file.open(fileName.toStdString());
638
639
        this->load(file);
640
        file.close();
641
         return 1;
642 }
643
644 char risk::get_number_of_player()
645
646
         return this->numberOfPlayers;
647
    }
648
649 void risk::set_number_of_player(char num )
650 {
651
        this->numberOfPlayers = num;
652
653
654
```

C:\risk_final\risk_final\Land.h

```
1 #pragma once
 2 #include "Player.h"
 3 #include "Country.h"
 4 class Land {
 5 public:
       Land();
 7
       Land& operator=(Land&);
       struct landInfo {
 8
           int bonus;
 9
10
           std::string reason;
11
       };
12
13
       Land(std::vector<std::string>, Land::landInfo);
14
15
       Land::landInfo calcTroops(Player&);
       Land::landInfo bonus;
16
17
18
       void save(std::ofstream& file) const;
19
       void load(std::ifstream& file, std::map<std::string, Player>& playersMap);
20 public:
       std::vector <Country> countries;
21
22 };
23
24
25 Land::landInfo& operator+=(Land::landInfo&, const Land::landInfo&);
```

Page 45

```
1 #include "Land.h"
 3 Land::Land()
 4 {
 5
        this->bonus = { 0, "no reason" };
 6
   }
 7
 8 Land::Land(std::vector<std::string> countries,Land::landInfo bonus)
 9
10
        this->bonus = bonus;
11
        int size = countries.size();
12
        this->countries.reserve(size);
13
14
        for (auto iter : countries) {
15
            this->countries.push_back(Country(iter));
16
        }
17 }
18
19 Land & Land::operator=(Land &src)
21
        // TODO: insert return statement here
22
        this->countries = src.countries;
23
        this->bonus = src.bonus;
24
        return *this;
25 }
26
27
28 Land::landInfo Land::calcTroops(Player & player)
29 {
        landInfo res = { 0,"" };
30
31
        for (auto country : this->countries) {
32
            if (country.getOwner().getName() == player.getName())
33
                res.bonus++;
34
        if (res.bonus == this->countries.size()) {
35
36
            res.bonus += this->bonus.bonus*3;
37
            res.reason += this->bonus.reason;
38
39
        return res;
40 }
41
   void Land::save(std::ofstream& file) const {
43
        for (auto country : this->countries) {
44
            country.save(file);
45
        }
46 }
47
   void Land::load(std::ifstream & file, std::map<std::string, Player>&
      playersMap)
49
50
        for (int i = 0; i < this->countries.size();i++) {
51
52
            this->countries[i].load(file);
53
            if (countries[i].getOwner().getName() == "None")
54
                return;
55
            Player tmp = playersMap[countries[i].getOwner().getName()];
```

```
C:\risk_final\risk_final\Land.cpp
56
           countries[i].setOwner(tmp);
57
           countries[i].incTroops(0);
58
       }
59
60 }
61
62 Land::landInfo & operator+=(Land::landInfo & lhs, const Land::landInfo & rhs)
63 {
       lhs.reason += "\n" + rhs.reason;
64
       lhs.bonus += rhs.bonus;
65
66
       return lhs;
67 }
68
```

C:\risk_final\risk_final\Country.h

```
1 #pragma once
 2 #include "Player.h"
 3 #include <QtWidgets/QMainWindow>
 4 #include "ui_my_risk.h"
 5 #include <string.h>
 6 #include <functional>
 7 #include <fstream>
 8 class Country :public QMainWindow {
 9
       Q_OBJECT
10 public:
      void setButton(QPushButton*);
11
12
       static Country* chosen;
       static Country* chosen2;
13
14
15
       static void setUI(Ui::my_riskClass*);
16
       Country(const std::string, Player player = Player());
17
       Country(const Country &);
18
       Country();
19
       Country& operator=(Country&);
20
21
       void setOwner(const Player&);
22
       void incTroops(int);
23
24
       bool validPath(const std::string, std::map<std::string, bool>& seen =
         std::map<std::string, bool>());
25
       Player getOwner() const;
26
       std::string getName() const;
27
       int getTroops();
28
29
       bool isNear(std::string) const;
30
       void setNehibors(std::vector<Country*>);
31
32
       void save(std::ofstream& file) const;
33
       void load(std::ifstream& file);
34 public slots:
35
      void init();
36 private:
37
       static Ui::my_riskClass* ui; //class member
38
       int troops;
39
       std::string name;
40
       Player owner;
41
       OPushButton* button;
42
       std::vector<Country*> nehibors;
43
44 };
```

```
1 #include "Country.h"
 2 #include <iostream>
 3 #include <qpushbutton.h>
 4 Country* Country::chosen;
 5 Country* Country::chosen2;
 6 void Country::init()
 7 {
 8
        if (!this->troops) {
           this->ui->textEdit->setText(QString(this->name.c_str()) + QString(" is >
 9
              a FREE country!"));
10
        }
        else
11
           this->ui->textEdit->setText(QString("ocupied") + QString(" by player: →
12
              ") + QString(this->owner.getName().c_str()) + QString(" with: ") +
              QString(std::to_string(this->troops).c_str()));
13
14
        Country::chosen2 = Country::chosen;
15
        Country::chosen = this;
16
17 }
18
19
20 Country::Country(const std::string name, Player player)
21 {
22
       this->name = name;
        this->troops = 0;
23
24
        this->owner = player;
25
26 }
27
28 Country::Country(const Country & src)
29 {
30
        this->troops = src.troops;
31
        this->name = src.name;
        this->owner = src.owner;
32
33
        this->button = src.button;
34 }
35
36
37 Country::Country()
38 {
39
        //this->button = 0;
40 }
41
42 Country & Country::operator=(Country &src)
43 {
44
        // TODO: insert return statement here
45
        this->troops = src.troops;
46
       this->name = src.name;
47
        this->owner = src.owner;
48
        this->button = src.button;
49
        return *this;
50 }
51
52 void Country::setButton(QPushButton* button) {
53
       this->button = button;
```

```
C:\risk_final\risk_final\Country.cpp
 54
    }
 55
 56 void Country::setOwner(const Player& owner)
 57 {
 58
         this->owner = owner;
 59
         QString style = QString("background-color:") + this->owner.getColor
           ().c_str();
 60
         this->button->setStyleSheet(style);
 61 }
 62
 63
 64 void Country::save(std::ofstream& file) const {
         std::regex re("\\s+");
 66
         std::string name = std::regex_replace(this->name, re, "_");
         file << name << "\n";
 67
         file << this->troops << "\n";</pre>
 68
 69
         file << this->owner.getName() << "\n";</pre>
 70
         file << "----\n";
 71 }
 72
 73 void Country::load(std::ifstream & file)
 74 {
 75
         file >> this->name;
 76
         std::string tmp;
 77
         file >> tmp;
        this->incTroops(std::stoi(tmp));
 78
 79
         std::string ownerName;
 80
        file >> ownerName;
 81
         this->owner.setName(ownerName);
 82
         Country::chosen = this;
 83
         std::string dummy;
 84
         file >> dummy;
 85 }
 86
    void Country::incTroops(int add)
 87
 88
         this->troops += add;
 89
 90
         if(add !=0)
 91
             this->button->setText(QString(std::to_string(this->troops).c_str()));
 92 }
 93
    bool Country::validPath(const std::string dest, std::map<std::string,bool>&
       seen)
 95 {
 96
         for (int i = 0; i < this->nehibors.size(); i++)
 97
             if (this->nehibors[i]->name == dest)
 98
                 return true;
 99
100
         for (int i = 0; i < this->nehibors.size(); i++)
             if (seen.find(this->nehibors[i]->name) == seen.end()) {
101
                 if (this->nehibors[i]->owner.getName() == this->owner.getName()) {
102
103
                     seen[this->nehibors[i]->name] = true;
104
                     if (this->nehibors[i]->validPath(dest, seen))
105
                         return true;
106
                 }
107
             }
```

```
C:\risk_final\risk_final\Country.cpp
108
        return false;
109 }
110
111 Player Country::getOwner() const
113
        return this->owner;
114 }
115
116 std::string Country::getName() const
117 {
118
        return std::string(this->name);
119 }
120
121 int Country::getTroops()
122 {
123
        return this->troops;
124 }
125
126 bool Country::isNear(std::string name) const
        for (int i = 0; i < this->nehibors.size(); i++)
128
129
            if (this->nehibors[i]->name == name)
130
                return true;
```

131132

133 }134

136 { 137

138139

140

146

147 } 148

141 } 142 143 }

return false;

144 Ui::my_riskClass* Country::ui;

Country::ui = ui;

135 void Country::setNehibors(std::vector<Country*> vec)

this->nehibors.push_back(vec[i]);

this->nehibors.reserve(vec.size());
for (int i = 0; i < vec.size(); i++) {</pre>

145 void Country::setUI(Ui::my_riskClass* ui) {

C:\risk_final\risk_final\risk_final\Player.h

```
1 #pragma once
 2 #include "Card.h"
 3 #include <QtWidgets/QMainWindow>
 4 #include <fstream>
 5 #include <regex>
 6 class Player {
 7 public:
       Player(const std::string name = "default", const std::string color =
 8
         "None", int order = 0, int id = -1);
 9
       Player(const Player&);
10
       Player& operator=(const Player&);
11
12
       void cleanCards(std::vector<Card> cards);
13
14
       void addBonusTroops(const int);
15
       void setBonusTroops(const int);
16
       int getBonusTroops();
17
18
       std::string getName() const;
19
       void setName(std::string);
20
       int getId() const;
21
       void SetOrder(int);
22
       int GetOrder();
       void show();
23
24
       std::string& getColor();
       void save(std::ofstream& file);
25
26
       void load(std::ifstream& file);
27
       Card addCard();
       std::vector<Card> getCards();
28
29 private:
30
       int bonusTroops;
31
       int order;
32
       std::vector<Card> cards;
33
       std::string name;
       std::string color;
34
35
       int id;
36 };
37
```

```
1 #include "Player.h"
 2
 3 Player::Player(const std::string name, const std::string color, int order,int →
 4
   {
 5
       this->name = name;
 6
       this->color = color;
 7
       this->order = order;
 8
       this->bonusTroops = -1;
 9
       this->id = id;
10 }
11
12 Player::Player(const Player &src):cards(src.cards)
13 {
        this->name = src.name;
14
15
        this->color = src.color;
16
        this->order = src.order;
17
        this->bonusTroops = src.bonusTroops;
18
        this->id = src.id;
19 }
20
21 Player & Player::operator=(const Player & src)
22 {
23
       this->order = src.order;
24
       this->color = src.color;
25
       this->name = src.name;
26
       this->cards = src.cards;
27
        this->bonusTroops = src.bonusTroops;
28
        this->id = src.id;
29
        return *this;
30 }
31
32 void Player::cleanCards(std::vector<Card> cards)
33 {
        for (int i = 0; i < cards.size(); i++) {</pre>
34
            auto it = std::find_if(this->cards.begin(), this->cards.end(), [&]
35
              (Card x) {return cards[i].getType() == x.getType(); });
36
            if (it != this->cards.end())
37
                this->cards.erase(it);
38
        }
39 }
40
41
42 void Player::addBonusTroops(const int bonus)
43
   {
44
        this->bonusTroops += bonus;
45 }
46
47 void Player::setBonusTroops(const int bonus)
48 {
49
        this->bonusTroops = bonus;
50 }
52 int Player::getBonusTroops()
53 {
54
        return this->bonusTroops;
```

```
55
     }
 56
 57
 58 std::string Player::getName() const
 59 {
 60
         return this->name;
 61 }
 62
 63 void Player::setName(std::string name)
 64 {
 65
         this->name = name;
 66 }
 67
 68 int Player::getId() const
 69 {
 70
         return this->id;
 71 }
 72
 73 std::string & Player::getColor()
 75
         // TODO: insert return statement here
 76
         return this->color;
 77
     }
 78
 79 void Player::save(std::ofstream& file)
 80 {
 81
         std::regex re("\\s+");
         file << std::regex_replace(this->name, re,"_") << "\n";</pre>
 82
 83
         file << this->id << "\n";</pre>
         file << this->order << "\n";</pre>
 84
         file << this->color << "\n";</pre>
 85
 86
         file << this->bonusTroops << "\n";</pre>
 87
         file << this->cards.size() << "\n";</pre>
 88
         for (Card card : cards)
 89
             card.save(file);
 90
         file << "----\n";
 91
 92 }
 93
 94 void Player::load(std::ifstream & file)
 95 {
 96
         file >> this->name;
 97
         std::string tmp;
 98
         file >> tmp;
 99
         this->id = std::stoi(tmp);
         file >> tmp;
100
         this->order = std::stoi(tmp);
101
102
         file >> this->color;
103
         file >> tmp;
104
         this->bonusTroops = std::stoi(tmp);
105
         int numberOfCards;
106
         file >> tmp;
107
         numberOfCards = std::stoi(tmp);
108
         this->cards.reserve(numberOfCards);
109
         for (int i = 0; i < numberOfCards; i++) {</pre>
110
             Card tmp;
```

```
C:\risk_final\risk_final\Player.cpp
111
            tmp.load(file);
112
            this->cards.push_back(tmp);
113
        }
114
        std::string dummy;
115
        file >> dummy;
116 }
117
118 Card Player::addCard()
119 {
120
        Card tmp;
121
        tmp.setRand();
122
        this->cards.push_back(tmp);
123
        return tmp;
124 }
125
126 std::vector<Card> Player::getCards()
127 {
128
        return this->cards;
129 }
130
131
132 void Player::SetOrder(int order )
133 {
134
        this->order = order;
135 }
136
137 int Player::GetOrder()
138 {
139
        return this->order;
140 }
141
142 void Player::show()
143 {
144
        for (int i = 0; this->cards.size(); ++i)
145
            this->cards[i].update();
146 }
147
```

```
1 #pragma once
 2 #include <QtWidgets/QMainWindow>
 3 #include <QWidget>
 4 #include <fstream>
 5 #include "Qdebug.h"
 6 #include "ui_my_risk.h"
 7 #define _numberOfValidCards_ 5
 8 class Card {
 9 public:
10
       static void nextBonus();
11
       static int getBonus();
       enum cardType { Joker, Solider, Cannon, Horse, Default };
12
13
       Card(cardType cardType = cardType::Default);
14
       Card(const Card&);
15
       Card& operator=(const Card&);
16
       bool operator==(const Card&) const;
17
       bool operator!=(const Card&) const;
       void setRand();
18
19
       void setButton(QPushButton*);
20
       void update();
21
       void save(std::ofstream& file) const;
22
       void load(std::ifstream& file);
23
       int maxTypes() const;
24
       int getType() const;
25
       void clean();
26
       bool isClicked() const;
27 private:
       std::string validType[_numberOfValidCards_];
28
29
       cardType type;
30
       QPushButton* button;
31
       bool clicked;
32
       static int bonus1;
33
       static int bonus2;
34
       static int bonus3;
35 public slots:
       void click();
36
37 };
38
```

```
1 #include "Card.h"
 2 int Card::bonus1;
 3 int Card::bonus2;
4 int Card::bonus3;
 5 void Card::nextBonus()
 6 {
 7
        Card::bonus1 = Card::bonus2 + Card::bonus3;
8
        Card::bonus2 = Card::bonus1;
 9
        Card::bonus3 = Card::bonus2;
10
11 }
12
13 int Card::getBonus()
14 {
        if (Card::bonus1 == 0) {
15
16
            Card::bonus1 = 3;
17
            Card::bonus2 = 2;
18
            Card::bonus3 = 1;
19
        }
20
        return Card::bonus1;
21
   }
22
23
   Card::Card(cardType type) : validType
                                                                                     ₽
      { "Joker", "Solider", "Cannon", "Horse", "Default" }
24
25
        this->type = type;
26
        this->clicked = false;
27 }
28
   Card::Card(const Card & src) : validType
29
                                                                                     ₽
      { "Joker", "Solider", "Cannon", "Horse", "Default" }
30
31
        this->type = src.type;
32
        this->clicked = src.clicked;
33 }
34
35 Card & Card::operator=(const Card & src)
36 {
37
        this->type = src.type;
38
        this->clicked = src.clicked;
39
        return *this;
40 }
41
42
   bool Card::operator==(const Card & rhs) const
43
   {
44
        if (this->type == Joker || rhs.type == Joker)
45
           return true;
46
47
        return ((int)this->type == (int)rhs.type);
48
   }
49
50 bool Card::operator!=(const Card & rhs) const
51 {
52
        if (this->type == Joker || rhs.type == Joker)
53
            return true;
54
```

```
C:\risk_final\risk_final\Card.cpp
 55
         return !((int)this->type == (int)rhs.type);
 56 }
 57
 58 void Card::setRand()
 59 {
 60
         int num = rand() % ((_numberOfValidCards_ - 1)*3 + 2);
         if (num != (_numberOfValidCards_ - 1) * 3 + 1) {    //chance of 1 to
 61
           (\_numberOfValidCards\_ - 1)*3 and every other card chance of 1 to
           (_numberOfValidCards_ - 1)
 62
            num = num % 3;
 63
         }
 64
         else {
 65
            num = 0;
 66
         this->type = Card::cardType(num);
 67
                                                                                     P
           { "Joker", "Solider", "Cannon", "Horse" };
 68
    }
 69
 70 void Card::setButton(QPushButton * button)
 72
         this->button = button;
 73
 74
    }
 75
 76 void Card::update()
 77 {
 78
         //QPixmap pic("Resources/cards/card"+ QString(this->validType[this-
           >type].c_str()) + ".PNG");
         QString pic = "border-image:url(:/new/Resources/cards/card" + QString
 79
           (this->validType[this->type].c_str()) + ".PNG);";
 80
         //this->button->setIcon()
 81
         this->button->setStyleSheet(pic);
 82
 83
         this->button->show();
 84
 85
    }
 86
 87 void Card::click()
 88 {
         if (this->type == Card::Default)
 89
 90
            return;
 91
 92
         QString pic;
93
         if (this->clicked) {
 94
             this->clicked = false;
            pic = "border-image:url(:/new/Resources/cards/card" + QString(this-
 95
               >validType[this->type].c_str()) + ".PNG);";
 96
         }
 97
         else {
 98
            this->clicked = true;
            pic = "border-image:url(:/new/Resources/cards/card" + QString(this-
 99
               >validType[this->type].c_str()) + "Shadow.PNG);";
100
         }
101
102
         //this->button->setIcon()
103
         this->button->setStyleSheet(pic);
```

```
104
    }
105
106 void Card::save(std::ofstream & file) const
107 {
108
         file << this->validType[this->type] << "\n";</pre>
109 }
110
111 void Card::load(std::ifstream & file)
112 {
113
         std::string tmp;
114
         file >> tmp;
         if (tmp == "Solider")
115
            this->type = Solider;
116
         else if (tmp == "Cannon")
117
118
             this->type = Cannon;
         else if (tmp == "Horse")
119
120
             this->type = Horse;
         else if (tmp == "Jokery;")
121
122
             this->type = Joker;
123
         else
124
             this->type = Default;
125
    }
126
127 int Card::maxTypes() const
128 {
129
         return _numberOfValidCards_;
130 }
131
132 int Card::getType() const
133 {
134
         return int(this->type);
135 }
136
137 void Card::clean()
138 {
139
         QString pic = "border-image:url(:/new/Resources/cards/card" + QString
           (this->validType[Card::Default].c_str()) + ".PNG);";
140
         this->button->setStyleSheet(pic);
141
         this->clicked = false;
142
         this->type = Default;
143
         this->button->show();
144 }
145
146 bool Card::isClicked() const
147
148
         return this->clicked;
149 }
150
```

C:\risk_final\risk_final\Card.cpp

```
1 #pragma once
 2 #include <QWidget>
 3 #include "ui_menu.h"
 4 #include <string>
 5 #include <iostream>
 7 class Cube {
 8 private:
       QLabel* label;
 9
10
       static std::string path;
11
       int rand_value;
12 public:
13
       Cube();
       void choosePic(int);
14
15
       void setValue(int);
       int getValue() const;
16
17
       static void setPath(std::string);
18
19
20
       void setLabel(QLabel*);
21
       void setPic(std::string);
22
       void random();
       void hide();
23
24
        void show();
25
        bool operator<(const Cube&) const;</pre>
        bool operator>(const Cube&) const;
26
        bool operator==(const Cube&) const;
27
28
29 };
```

```
1 #include "Cube.h"
 2 #include "Vector.h"
 3 std::string Cube::path;
 4 void Cube::choosePic(int num)
 5 {
 6
 7
       QPixmap pic(QString(this->path.c_str()) + QString(num + '0') + ".png");// →
          ("Resources/cubes/1.png");
 8
       label->setPixmap(pic);
 9
       label->show();
10 }
11 void Cube::setValue(int rand val)
12 {
13
       this->rand_value = rand_val;
14 }
15 int Cube::getValue() const
16 {
17
       return this->rand_value;
18 }
19 Cube::Cube()
20 {
21
       this->rand_value = -1;
22 }
23
24 void Cube::setPath(std::string path)
25 {
26
       Cube::path = path;
27 }
28
29
30 void Cube::setLabel(QLabel * label)
31 {
32
       this->label = label;
33
       this->label->setVisible(true);
34 }
35
36 void Cube::setPic(std::string str)
37
38
39
       QPixmap pic(str.c_str());//("Resources/cubes/1.png");
40
       label->setPixmap(pic);
41
42
       label->show();
43 }
44
45 void Cube::random()
46 {
47
       this->rand_value = std::rand() % 6 + 1;
48
       this->choosePic(this->rand_value);
49 }
50
51 void Cube::hide()
52 {
53
       {
           this->label->setVisible(false);
54
55
       }
```

```
C:\risk_final\risk_final\Cube.cpp
56 }
57
58 void Cube::show()
59 {
60
       this->label->setVisible(true);
61 }
62
63 bool Cube::operator<(const Cube & rhs) const
65
       return this->rand_value < rhs.rand_value;</pre>
66 }
67
68 bool Cube::operator>(const Cube & rhs) const
69 {
70
       return this->rand_value > rhs.rand_value;
71 }
72
73 bool Cube::operator==(const Cube & rhs) const
74 {
75
       return this->rand_value == rhs.rand_value;
76 }
```

77

Stage.h

C:\risk_final\risk_final\Stage.h

```
1
```

```
1 #pragma once
 2 #include <QtWidgets/QMainWindow>
 3 #include "ui_my_risk.h"
 4 #include "Player.h"
 5 #include "Country.h"
 6 #include "Cube.h"
 7 #include "Move.h"
 8 class Stage {
 9 public:
10
       Stage(Ui::my_riskClass&, Player& player, Move& moveUi);
11
       virtual void preAction()=0;
12
       virtual void action()=0;
13
       virtual void afterAction()=0;
14
       virtual void plus(int sel = 0);
15
       virtual void minus(int sel = 0);
       void showTroops();
16
17
       bool getNexEn();
18
      void setNextEn(const bool en);
19
       bool getEnterEn();
20
       void setEnterEn(const bool en);
       void setPlayer(Player&);
21
22 protected:
23
       int getBonusTroops(int sel = 0);
24
       bool isMine(int sel = 0);
25
26 protected:
       Ui::my_riskClass& ui;
27
       Move& moveUi;
28
29
       Player& player;
30
       bool nextEn;
31
       bool EnterEn;
32
33
34
35 };
36
37
```

```
1 #include "Stage.h"
 2
 3 Stage::Stage(Ui::my_riskClass& ui,Player& player, Move& moveUi): ui(ui),player →
      (player), moveUi(moveUi)
 4
 5
        this->nextEn = false;
 6
        this->EnterEn = true;
 7
   }
 8
 9 bool Stage::getNexEn()
10 {
11
        return this->nextEn;
12 }
13
14 void Stage::setNextEn(const bool en)
15
   {
16
        this->nextEn = en;
17 }
18
19 bool Stage::getEnterEn()
20 {
21
        return this->EnterEn;
22
   }
23
24 void Stage::setEnterEn(const bool en)
25 {
26
        this->EnterEn = en;
27 }
28
29 void Stage::setPlayer(Player & player)
30 {
31
        this->player = player;
32 }
33
34 void Stage::showTroops()
35 {
        int bonus = this->player.getBonusTroops();
36
37
        QString text = std::to_string(bonus).c_str();
38
        this->ui.troopsP->setText(text);
39 }
40 int Stage::getBonusTroops(int sel)
41 {
42
        if (sel == 0) {
43
            std::string text = this->ui.troopsP->toPlainText().toStdString();
44
            return std::stoi(text);
45
        }
        else if (sel == 1) {
46
47
            std::string text = this->ui.troopsP_2->toPlainText().toStdString();
48
            return std::stoi(text);
49
        }
50
        return -1;
51
   }
52
53 bool Stage::isMine(int sel)
54 {
55
```

100

PlaceMap.h

C:\risk_final\risk_final\PlaceMap.h

```
1 #pragma once
 2 #include "Stage.h"
3 #include "Land.h"
 4 #include "Player.h"
 5 #include "Country.h"
 7 class PlaceMap : public Stage {
 8 public:
9
       PlaceMap(Ui::my_riskClass&, Player& player, Move& moveUi, int& counter);
10
11
       void action();
12
       void preAction();
13
       void afterAction();
14 private:
15
       int& counter;
16 };
```

PlaceMap.cpp

C:\risk_final\risk_final\risk_final\PlaceMap.cpp

```
1 #include "PlaceMap.h"
 3 PlaceMap::PlaceMap(Ui::my_riskClass & ui, Player & player, Move& moveUi , int&
     counter) : Stage(ui, player, moveUi), counter(counter)
 4
 5
   }
 6
 7
   void PlaceMap::action()
 8
 9
       if (this->isMine() && Country::chosen->getTroops()) {
10
           this->ui.textEdit->setText(QString("At this stage you need to fill-up >
             the map .."));
11
           this->EnterEn = false;
12
            return;
13
       }
14
15
       if (Country::chosen->getTroops()) {
16
           this->ui.textEdit->setText(QString("I see you got NA education .."));
17
            this->EnterEn = false;
18
            return;
19
       }
20
       this->EnterEn = true;
21
       Country::chosen->setOwner(this->player);
22
       Country::chosen->incTroops(1);
23
       this->counter++;
       this->ui.textEdit->setText(QString("initial Country: ") + QString
24
          (Country::chosen->getName().c_str()) + QString(" by player: ") + QString →
          (this->player.getName().c_str()));
25
       this->player.addBonusTroops(-1);
26
27 }
28
29 void PlaceMap::preAction()
30 {
31
       this->showTroops();
32 }
33
34 void PlaceMap::afterAction()
35 {
36
       this->showTroops();
37 }
38
```

PlaceTropps.h

C:\risk_final\risk_final\PlaceTroops.h

```
1 #pragma once
2 #include "Stage.h"
3 #include "Land.h"
 4 #include "Player.h"
 5 #include "Country.h"
 7 class PlaceTroops : public Stage {
 8 public:
9
       PlaceTroops(Ui::my_riskClass&, Player& player, Move& moveUi,int& counter);
10
11
      void action();
12
      void preAction();
13
      void afterAction();
14
      void showInfo();
15
16 private:
17
       int& counter;
18
19 };
```

Page 68

PlaceTroops.cpp

C:\risk_final\risk_final\risk_final\PlaceTroops.cpp

```
1 #include "PlaceTroops.h"
 3 PlaceTroops::PlaceTroops(Ui::my_riskClass & ui, Player & player, Move &
                                                                                     P
     moveUi,int& counter) : Stage(ui, player, moveUi), counter(counter)
 4
 5
   }
 6
 7
   void PlaceTroops::action()
 8
 9
        if (!this->isMine())
10
            this->ui.textEdit->setText(QString(Country::chosen->getName().c str()) >
11
              + QString(" isn't yours"));
12
            this->EnterEn = false;
13
            return;
14
        }
15
       this->EnterEn = true;
16
        this->counter++;
17
        this->ui.textEdit->setText(QString("add one troop at") + QString
          (Country::chosen->getName().c_str()) + QString(" to ") + QString(this-
          >player.getName().c_str()));
18
        Country::chosen->incTroops(1);
19
        this->player.addBonusTroops(-1);
20 }
21
22
23 void PlaceTroops::preAction()
24 {
25
        this->showInfo();
26
        this->showTroops();
27
28
29 void PlaceTroops::afterAction()
30 {
31
32 }
33
34 void PlaceTroops::showInfo()
35 {
36
        QString picPath = QString("Resources/players/") + QString(this-
          >player.getColor().c str()) + ".png";
37
        QPixmap pic(picPath);//("Resources/cubes/1.png");
38
        this->ui.colorP->setPixmap(pic);
39
        this->ui.colorP->show();
40 }
41
```

C:\risk_final\risk_final\Draft.h

```
1 #pragma once
2 #include "Stage.h"
 3 #include "Land.h"
4 #include "Player.h"
5 #include "Country.h"
 6 class Draft : public Stage {
7 public:
       Draft(Ui::my_riskClass&, Player& player, Move& moveUi, std::vector<Card>& >
 8
         cards,int _maxCardsPerPlayer);
9
10
       void setLands(std::vector<Land*> lands);
       void action();
11
       void preAction();
12
       void afterAction();
13
14
       void showBonusTroops(std::string);
15
       Land::landInfo calcTroops();
16
      void plus(int sel = 0);
17
       void minus(int sel = 0);
18
       void convert();
19 private:
20
      int newCard;
       std::vector<Card>& cards;
21
22
       std::vector<Land*> lands;
23
       int _maxCardsPerPlayer;
24
25 };
```

```
#include "Draft.h"
 1
 3 Draft::Draft(Ui::my_riskClass & ui, Player& player, Move& moveUi,
                                                                                     P
      std::vector<Card>& cards, int _maxCardsPerPlayer_) : Stage(ui, player,
      moveUi), cards(cards)
 4
   {
 5
        this->cards = cards;
        this->_maxCardsPerPlayer = _maxCardsPerPlayer_;
 6
 7
        this->ui.textEdit->setText(QString("welcome to stage draft you need to
          place all your troops in this stage:"));
 8 }
 9
10 void Draft::setLands(std::vector<Land*> lands)
11 {
12
        this->lands = lands;
13
14 void Draft::action()
15 {
16
        if (!this->isMine() || this->EnterEn == false)
17
18
            return;
19
20
        this->ui.textEdit->setText(
21
            "add" + QString(std::to_string(this->getBonusTroops()).c_str()) + "
              number of troops at: '
            + Country::chosen->getName().c_str() + " to: " + this->player.getName →
22
              ().c str());
23
        Country::chosen->incTroops(this->getBonusTroops());
24
        this->player.addBonusTroops((-1)*this->getBonusTroops());
25
        this->showTroops();
        if (this->player.getBonusTroops() == 0) {
26
27
            this->nextEn = true;
28
            this->EnterEn = false;
29
        }
30
   }
31
32 void Draft::preAction()
33 {
34
        Land::landInfo info = this->calcTroops();
35
        if (info.bonus == 0)
36
        {
37
            this->nextEn = true;
38
39
        this->player.setBonusTroops(info.bonus);
40
        this->showBonusTroops(info.reason);
41
        std::vector<Card> cards = this->player.getCards();
42
        for (int i = 0; i < 5; i++) //todo const</pre>
43
            if (i < cards.size())</pre>
44
                this->cards[i] = cards[i];
45
            else
46
                this->cards[i] = Card();
47
48
        for (int i = 0; i < _maxCardsPerPlayer; i++)</pre>
49
            this->cards[i].update();
50 }
51
```

104 { 105

bool force = false;

```
C:\risk_final\risk_final\Draft.cpp
106
         if (sel >= 10) {
107
             force = 1;
108
             sel %= 10;
109
         if (sel == 0) {
110
             int number = this->getBonusTroops() - 1;
111
             if (number > 0 || force)
112
113
                 this->ui.troopsP->setText(std::to_string(number).c_str());
114
             else
115
                 this->ui.textEdit->setText("I see you got NA education ...");
116
         }
117
         else if (sel == 1) {
             int number = this->getBonusTroops(1) - 1;
118
119
             if (number > 0 || force)
120
                 this->ui.troopsP_2->setText(std::to_string(number).c_str());
121
             else
122
                 this->ui.textEdit->setText("I see you got NA education ...");
123
         }
124 }
125
126 void Draft::convert()
127 {
128
         std::vector<Card> cards;
129
         for (int i = 0; i < this->cards.size(); i++) { //3 same type
130
             if (this->cards[i].getType() != Card::Default && this->cards
               [i].isClicked()) {
131
                 cards.push_back(this->cards[i]);
132
             }
133
         }
         if (cards.size() != 3) {
134
135
             this->ui.textEdit->setText("no matching cards for convert! choose 3
               valid cards! ");
136
             return;
137
         }
138
         if (cards[0] == cards[1] && cards[0] == cards[2]) {
139
             //check tripple of the same type / joker type.
140
141
             for (int i = 0; i < cards.size(); i++) {</pre>
142
                 auto it = std::find_if(this->cards.begin(), this->cards.end(), [&] →
                   (Card x) {return cards[i].getType() == x.getType(); });
                 if (it != this->cards.end())
143
                     it->clean();
144
145
             this->player.cleanCards(cards);
146
147
             int bonus = Card::getBonus();
148
             this->player.addBonusTroops(bonus);
149
             Card::nextBonus();
150
             this->ui.troopsP->setText(std::to_string(this->player.getBonusTroops
               ()).c str());
151
             this->ui.textEdit->setText("N1! get bonus of " +
152
                 QString(std::to_string(bonus).c_str())+
                 " troops!");
153
             this->EnterEn = true;
154
155
         else if(cards[0] != cards[1] && cards[0] != cards[2] && cards[1] != cards →
156
           [2]){
```

```
C:\risk_final\risk_final\Draft.cpp
157
             //check tripple of diffrent types.
158
             for (int i = 0; i < cards.size(); i++) {</pre>
159
                 auto it = std::find_if(this->cards.begin(), this->cards.end(), [&] >
                   (Card x) {return cards[i].getType() == x.getType(); });
160
                 if (it != this->cards.end())
161
                     it->clean();
162
             }
             this->player.cleanCards(cards);
163
             int bonus = Card::getBonus();
164
165
             this->player.addBonusTroops(bonus);
166
             Card::nextBonus();
             this->ui.troopsP->setText(std::to_string(bonus).c_str());
167
168
             this->EnterEn = true;
169
         }
170
         else {
             this->ui.textEdit->setText("no matching cards for convert! choose 3
171
               valid cards! ");
172
             return;
173
         }
174
175
176
177
178 }
179
```

```
1
```

```
#include "Attack.h"
 1
 2
   Attack::Attack(Ui::my_riskClass & ui, Player& player, Move& moveUi, Card*
      card) : Stage(ui,player,moveUi)
 4
   {
 5
        this->newCard = false;
 6
        this->card = card;
 7
        this->setNextEn(true);
 8
        this->ui.TROOPS_2->setVisible(true);
 9
        this->boom = nullptr;
10 }
11
12 void Attack::action()
13 {
14
15
        if (this->getBonusTroops() == 0 || this->getBonusTroops(1) == 0) {
16
            this->ui.textEdit->setText("I see you got NA education .. choose
17
              number of troops!");
18
            return;
19
        }
20
21
22
        //chosen2 attaker
23
        //chosen defender
        if (!this->isMine(1))//attacker is not me.
24
25
            return;
26
        if (this->isMine()) { //country to attack is mine.
27
            this->ui.textEdit->setText("I see you got NA education .. you cannot
              attack yourself!");
28
            return;
29
        if (!Country::chosen2->isNear(Country::chosen->getName())) {
30
                                                                         //cannot
          attack un-near countries.
            this->ui.textEdit->setText("they are not near!");
31
            return;
32
        }
33
34
35
        //troops and countries are valid
36
37
        //attack logic --
        //make cubes according to troops.
38
39
        //random values.
40
        std::vector<Cube> attacker;
41
        std::vector<Cube> defender;
42
43
        attacker.reserve(this->getBonusTroops());
44
        for (int i = 0; i < this->getBonusTroops(); i++)
45
            attacker.push back(Cube());
46
        attacker[0].setLabel(this->ui.cubeP_1);
47
        if (attacker.size() > 1)
            attacker[1].setLabel(this->ui.cubeP 2);
48
49
        if (attacker.size() > 2)
50
            attacker[2].setLabel(this->ui.cubeP_3);
51
52
        defender.reserve(this->getBonusTroops(1));
```

Attack.cpp

```
C:\risk_final\risk_final\Attack.cpp
 53
         for (int i = 0; i < this->getBonusTroops(1); i++)
 54
             defender.push back(Cube());
 55
         defender[0].setLabel(this->ui.cubeP2_1);
 56
         if (defender.size() > 1)
             defender[1].setLabel(this->ui.cubeP2_2);
 57
 58
 59
         for (int i = 0; i < attacker.size(); i++)</pre>
 60
 61
             attacker[i].random();
 62
         for (int i = 0; i < defender.size(); i++)</pre>
 63
             defender[i].random();
 64
         //std::sort(attacker.begin(), attacker.end(), [](const Cube& x, const
 65
           Cube& y) {return x.get_rand_value() > y.get_rand_value(); });
         //std::sort(defender.begin(), defender.end(), [](const Cube& x, const
 66
           Cube& y) {return x.get_rand_value() > y.get_rand_value(); });
 67
         std::sort(attacker.begin(), attacker.end());//from small to big
 68
         std::sort(defender.begin(), defender.end());
 69
         int cnt = 0;
         if (std::min(attacker.size(), defender.size()) == 1)
                                                                 //is 1 troop do
 70
           the loop only once.
 71
             cnt = 1;
 72
         do {
 73
             cnt++;
 74
             if (attacker.back() > defender.back()) {    //attaker have to biggest >
 75
                 auto it = std::find if(attacker.begin(), attacker.end(), [&](const →
                    Cube& x) {return x > defender.back(); }); //go from small to →
                   big.
                 //del both cubes.
 76
 77
                 attacker.erase(it);
 78
                 defender.pop_back();
 79
 80
                 Country::chosen->incTroops(-1);
                                                     //erase from defender!
 81
                 this->minus(11);
             }
 82
             else {
                                                     //else
 83
 84
                 auto it = std::find_if(defender.begin(), defender.end(), [&](const →
                    Cube& x) {return (!(attacker.back() > x)); }); //go from small →
                   to big.
 85
                 //del both cubes.
 86
                 defender.erase(it);
                 attacker.pop_back();
 87
 88
 89
                 Country::chosen2->incTroops(-1);
                                                          //erase from attaker!
 90
                 this->minus(10);
 91
         } while (this->getBonusTroops() && this->getBonusTroops(1) && cnt != 2);
 92
               //end if 2 loops or troops is empty.
 93
 94
 95
         if (Country::chosen->getTroops())
 96
             return:
 97
         this->boom= new QMediaPlayer();
         this->boom->setMedia(QUrl("qrc:/new/Resources/Explosion.wav"));
 98
 99
         this->boom->play();
```

```
C:\risk_final\risk_final\Attack.cpp
100
         this->newCard = true;
101
         Country::chosen->setOwner(this->player);
                                                 //send to defender
102
         this->moveUi.setTo(Country::chosen);
103
         this->moveUi.setFrom(Country::chosen2); //send From attaker
104
         this->moveUi.setMax(Country::chosen2->getTroops() - 1);
         this->moveUi.setMin(this->getBonusTroops());
105
106
         this->moveUi.setFocus();
107
         this->moveUi.show();
108 }
109
110 void Attack::preAction()
111 {
         this->ui.textEdit->setText("Please choose a country to attack and the
112
           number of troops to send");
         this->ui.troopsP->setText("1");
113
114
         this->ui.troopsP_2->setText("1");
115 }
116
117 void Attack::afterAction()
118 {
119
                                 //if new card update gui and player.
         if (this->newCard) {
120
             this->newCard = false;
121
             Card tmp = this->player.addCard();
122
             *this->card = tmp;
123
             this->card->update();
124
125
         this->ui.cubeP 1->hide();
126
         this->ui.cubeP_2->hide();
127
         this->ui.cubeP_3->hide();
128
         this->ui.cubeP2_1->hide();
129
         this->ui.cubeP2_2->hide();
130
         this->ui.cubeP2 3->hide();
131
         if (this->boom)
132
         {
133
             delete boom;
134
         this->ui.TROOPS_2->setVisible(false);
135
136
    }
137
138 void Attack::plus(int sel)
139 {
         if (sel == 0) {
140
141
             int number = this->getBonusTroops() + 1;
                 if (number > 3 || number >= Country::chosen2->getTroops())
142
143
                     this->ui.textEdit->setText("too many troops! sorry ..");
144
                 else
                     this->ui.troopsP->setText(std::to_string(number).c_str());
145
146
         }
147
         else if (sel == 1) {
148
             int number = this->getBonusTroops(1) + 1;
149
150
                 if (number > 2 || number > Country::chosen->getTroops())
                     this->ui.textEdit->setText("too many troops! sorry ..");
151
152
                 else
153
                     this->ui.troopsP_2->setText(std::to_string(number).c_str());
154
         }
```

```
C:\risk_final\risk_final\Attack.cpp
```

```
4
```

```
155
    }
156
157 void Attack::minus(int sel)
158 {
159
        bool force = false;
160
         if (sel >= 10) {
             force = 1;
161
             sel %= 10;
162
         }
163
164
         if (sel == 0) {
165
             int number = this->getBonusTroops() - 1;
                 if (number > 0 || force)
166
167
                     this->ui.troopsP->setText(std::to_string(number).c_str());
168
                 else
                     this->ui.textEdit->setText("I see you got NA education ...");
169
170
         }
171
         else if (sel == 1) {
             int number = this->getBonusTroops(1) - 1;
172
173
                 if (number > 0 || force)
174
                     this->ui.troopsP_2->setText(std::to_string(number).c_str());
175
                 else
176
                     this->ui.textEdit->setText("I see you got NA education ...");
177
         }
178 }
179
```

Reinforcement.h

${\tt C:\risk_final\risk_final\risk_final\Reinforcement.h}$

```
1 #pragma once
2 #include "Stage.h"
3 #include "Land.h"
 4 #include "Player.h"
5 #include "Country.h"
 7 class Reinforcement : public Stage {
 8 public:
       Reinforcement(Ui::my_riskClass& ui, Player& player, Move& moveUi);
9
10
11
      void action();
      void preAction();
12
      void afterAction();
13
14
15
16
17 };
```

Reinforcement.cpp

C:\risk_final\risk_final\risk_final\Reinforcement.cpp

1 #include "Reinforcement.h" Reinforcement::Reinforcement(Ui::my_riskClass & ui, Player & player, Move & moveUi) : Stage(ui, player, moveUi) 4 { 5 this->setNextEn(true); this->ui.TROOPS->setVisible(false); 6 } 7 8 9 void Reinforcement::action() 10 { if (Country::chosen->getOwner().getName() != Country::chosen2->getOwner 11 ().getName() || !this->isMine()) { 12 this->ui.textEdit->setText("Please choose 2 owned countries which are connected!"); 13 return; 14 } if (!Country::chosen2->validPath(Country::chosen->getName())) { 15 16 this->ui.textEdit->setText("Those countries are not connected!"); 17 return; 18 } if (Country::chosen2->getTroops() == 1) 19 20 21 this->ui.textEdit->setText("you cant move troops from country that have → inly 1 soilder!"); 22 return; 23 } 24 //pop up 25 26 this->moveUi.setTo(Country::chosen); //send to 27 this->moveUi.setFrom(Country::chosen2); //send From 28 this->moveUi.setMax(Country::chosen2->getTroops() - 1); 29 this->moveUi.setMin(1); 30 this->moveUi.setFocus(); this->moveUi.show(); 31 32 } 33 34 void Reinforcement::preAction() 35 { 36 37 } 39 void Reinforcement::afterAction() 40 { 41 this->ui.TROOPS->setVisible(true); 42 } 43

```
#pragma once
 2 #include "iostream"
 3 template <typename T>
 4 class Vector {
 5 public:
 6
        Vector();
 7
        typedef T * iterator;
 8
        Vector(const Vector<T>& v);
 9
        ~Vector();
10
        //geters
11
        unsigned int capacity() const;
12
        unsigned int size() const;
13
        bool empty() const;
14
       T& front();
15
       T& back();
16
       iterator begin();
        iterator end();
17
18
       void push_back(const T& value);
19
       void pop_back();
20
       void reserve(unsigned int capacity);
21
       T & operator[](unsigned int index);
22
       T & operator[](unsigned int index)const;
23
        void clear();
24
       Vector<T> & operator = (const Vector<T> &);
25 private:
        unsigned int size_;
26
27
        unsigned int capacity_;
28
        T* buffer;
29 };
30
31 template<class T>
32 Vector<T>::Vector() {
33
        capacity_ = 0;
34
        size = 0;
        buffer = nullptr;
35
36 }
37
38 template<class T>
39 Vector<T>::~Vector() {
40
        if (this->capacity_)
41
            delete[] buffer;
42 }
43 template<class T>
   Vector<T>::Vector(const Vector<T> & v) {
45
        size_ = v.size_;
46
        this->capacity_ = v.capacity_;
47
        if (this->buffer)
48
        {
49
            delete[] buffer;
50
        }
        buffer = new T[this->capacity_];
51
        for (unsigned int i = 0; i < size ; i++)</pre>
52
53
            buffer[i] = v.buffer[i];
54 }
55 template<class T>
56 void Vector<T>::reserve(unsigned int capacity) {
```

```
C:\risk_final\risk_final\risk_final\Vector.h
 57
         T * newBuffer = new T[capacity];
 58
         for (int i = 0; i < this->size_; i++)
 59
             newBuffer[i] = buffer[i];
 60
         this->capacity_ = capacity;
 61
         delete[] buffer;
         buffer = newBuffer;
 62
 63 }
 64
 65 template<class T>
 66 void Vector<T>::push_back(const T & v) {
         if (this->size_ >= this->capacity_) {
 67
 68
             if (!this->capacity )
 69
 70
                 this->capacity_ = 1;
 71
             }
 72
             reserve(2 * capacity_);
 73
         }
         buffer[size_++] = v;
 74
 75 }
 76
 77 template<class T>
 78 void Vector<T>::pop_back() {// at the end we delete vector at size capacity
         size_--;
 79
 80 }
 81 template<class T>
 82 T& Vector<T>::front() {
 83
         return buffer[0];
 84 }
 85
 86 template<class T>
    T& Vector<T>::back() {
 88
         return buffer[this->size_ - 1];
 89 }
 90
 91 template<class T>
 92 typename Vector<T>::iterator Vector<T>::end() {
 93
         return buffer + size_;
 94
    }
 95
 96 template<class T>
 97 typename Vector<T>::iterator Vector<T>::begin() {
 98
         return buffer;
 99 }
100 template <class T>
101 bool Vector<T>::empty() const {
102
         return size_ == 0;
103 }
104
105 template<class T>
106 Vector<T>& Vector<T>::operator = (const Vector<T> & v) {
107
         if (buffer)
108
             delete[] buffer;
109
         this->size_ = v.size_;
110
         this->capacity_ = v.capacity_;
111
         buffer = new T[this->capacity_];
112
         for (unsigned int i = 0; i < this->size_; i++)
```

```
113
            buffer[i] = v.buffer[i];
114
        return *this;
115 }
116 template<class T>
117 unsigned int Vector<T>::size() const {
        return size_;
119 }
120
121 template<class T>
122 T& Vector<T>::operator[](unsigned int index) {
      return buffer[index];
123
124 }
125 template<class T>
126  T& Vector<T>::operator[](unsigned int index)const {
127
        return buffer[index];
128 }
129
130 template<class T>
131 void Vector<T>::clear() {// at the end we delete vector at size capacity
        if (buffer != nullptr)
133
            delete[] buffer;
134
        this->size_ = 0;
135
        this->capacity_ = 0;
136 }
137
138 template<class T>
139 unsigned int Vector<T>:::capacity()const {
140
        return this->capacity_;
141 }
142
```

PlayerStats.h

C:\risk_final\risk_final\PlayerStats.h

```
1 #pragma once
 2 #include "Player.h"
 3 #include "PlayerStatWrap.h"
 4 #include <fstream>
 5 #include "Vector.h"
 6 class PlayerStats {
 7 public:
 8
       PlayerStats();
9
10
       template <typename T>
11
       struct Info {
12
           std::string owner;
13
           T value;
14
       };
15
       PlayerStats::Info<int> getMaxWins();
16
17
       PlayerStats::Info<int> getMaxLooses();
18
       PlayerStats::Info<int> getMaxTimePlayed();
19
       PlayerStats::Info<int> getMaxGamesPlayed();
20
       const Vector<PlayerStatWrap>& get_players();
       std::string getAvgTime();
21
22
       void loadStats(std::ifstream& file);
23
       void saveStats(std::ofstream& file);
24
       void addToStats(const PlayerStatWrap&);
25
       void clear();
26
       int size();
27 private:
       Vector<PlayerStatWrap> players;
28
29
       //std::vector<PlayerStatWrap> players;
30
       PlayerStats::Info<int> getMaxInt(int(*)(const PlayerStatWrap));
31
32 };
33
34
```

C:\risk_final\risk_final\risk_final\PlayerStats.cpp

```
1 #include "PlayerStats.h"
 2
 3
 4
  PlayerStats::PlayerStats()
 5 {
 6
 7
   }
 8
 9
   PlayerStats::Info<int> PlayerStats::getMaxWins()
10 {
        Info<int> res = { "" ,0 };
11
        for (int i = 0; i < this->players.size(); ++i) {
12
13
            if (this->players[i].get_wins() >= res.value) {
14
                res.value = this->players[i].get_wins();
15
                res.owner = this->players[i].get_name();
16
            }
17
        }
18
        return res;
19
   }
20
21
   void PlayerStats::loadStats(std::ifstream & file)
22
       while (file.eof() == false) {
23
24
            PlayerStatWrap res;
25
            res.loadStats(file);
            this->addToStats(res);
26
27
        }
28
29 }
30
31 void PlayerStats::saveStats(std::ofstream & file)
32 {
        for (int i = 0; i < this->players.size(); i++) {
33
34
            this->players[i].saveStats(file);
35
            if (i + 1 != this->players.size())
                file << "\n";</pre>
36
37
        }
38
   }
39
40
41 void PlayerStats::addToStats(const PlayerStatWrap& playerWrap)
42 {
43
        for (int i = 0; i < this->players.size(); i++) {
44
            if (this->players[i].get_name() == playerWrap.get_name()) {
45
                this->players[i] = this->players[i] + playerWrap;
46
                return;
47
            }
48
        }
49
        this->players.push_back(playerWrap);
50
   }
51
52 void PlayerStats::clear()
53 {
54
        this->players.clear();
55 }
56
```

```
57
 58 int PlayerStats::size()
 59 {
 60
         return this->players.size();
 61 }
 62
 63 PlayerStats::Info<int> PlayerStats::getMaxInt(int(*)(const PlayerStatWrap))
 64 {
 65
         return PlayerStats::Info<int>();
 66 }
 67
 68 PlayerStats::Info<int> PlayerStats::getMaxLooses()
 69 {
         Info<int> res = { "" ,0 };
 70
         for (int i = 0; i < this->players.size(); ++i) {
 71
 72
             if (this->players[i].get_looses() >= res.value) {
 73
                 res.value = this->players[i].get_looses();
 74
                 res.owner = this->players[i].get_name();
 75
             }
 76
         }
 77
         return res;
 78
    }
 79
 80 PlayerStats::Info<int> PlayerStats::getMaxTimePlayed()
 81 {
         Info<int> res = { "" ,0 };
 82
 83
         for (int i = 0; i < this->players.size(); ++i) {
 84
             if (this->players[i].get_timePlayed() >= res.value) {
 85
                 res.value = this->players[i].get_timePlayed();
                 res.owner = this->players[i].get_name();
 86
 87
             }
 88
         }
 89
         return res;
 90
    }
 91
 92 PlayerStats::Info<int> PlayerStats::getMaxGamesPlayed()
 93 {
         Info<int> res = { "" ,0 };
 94
 95
         for (int i = 0; i < this->players.size(); ++i) {
 96
             if (this->players[i].get_gamesPlayed() >= res.value) {
 97
                 res.value = this->players[i].get gamesPlayed();
 98
                 res.owner = this->players[i].get name();
 99
             }
100
         }
101
         return res;
102 }
103
104 const Vector<PlayerStatWrap>& PlayerStats::get_players()
105 {
106
         return this->players;
107
    }
108
109
110 std::string PlayerStats::getAvgTime()
111 {
112
         int time = 0;
```

PlayerStatsWarp.h

C:\risk_final\risk_final\PlayerStatWrap.h

```
1 #pragma once
 2 #include "Player.h"
 3 #include <chrono>
 4 #include <fstream>
 5 class PlayerStatWrap {
 6 public:
        PlayerStatWrap();
 7
 8
        PlayerStatWrap(const PlayerStatWrap& src);
 9
        PlayerStatWrap(const Player& player,int win, int elapsed_seconds);
10
        void loadStats(std::ifstream& file);
11
       void saveStats(std::ofstream& file);
12
        std::string get name() const;
        int get_timePlayed() const;
14
        int get_gamesPlayed() const;
15
        int get_wins() const;
16
        int get_looses() const;
        int get_id() const;
17
18
        PlayerStatWrap& operator+(const PlayerStatWrap& src);
19
        PlayerStatWrap& operator=(const PlayerStatWrap& src);
20 private:
21
        int timePlayed;
22
        int gamesPlayed;
23
        int wins;
24
       int looses;
25
       int id; //todo
26
       std::string name;
27
       //int maxTroops;
28
       //int maxCountries;
29
30 };
```

```
-
```

```
1 #include "PlayerStatWrap.h"
 2 //getline(fileHandle, org, ' ');
 3 PlayerStatWrap::PlayerStatWrap()
 4 {
 5
        this->timePlayed = 0;
 6
        this->gamesPlayed = 1;
        this->wins = 0;
 7
 8
        this->looses = 0;
 9
        this->id = 0;
10 }
11
12 PlayerStatWrap::PlayerStatWrap(const PlayerStatWrap & src)
13 {
14
        this->name = src.name;
15
        this->wins = src.wins;
16
        this->looses = src.looses;
17
        this->timePlayed = src.timePlayed;
18
        this->gamesPlayed = src.gamesPlayed;
19
        this->id = src.id;
20 }
21
22 PlayerStatWrap::PlayerStatWrap(const Player& player, int win, int
      elapsed seconds)
23 {
24
        this->name = player.getName();
25
        if (win == 0) {
26
            this->wins = 0;
27
            this->looses = 0;
28
        else if (win == 1) {
29
30
            this->wins = 1;
31
            this->looses = 0;
32
        }
33
        else if (win == -1) {
            this->wins = 0;
34
            this->looses = 1;
35
36
37
        this->timePlayed = elapsed seconds;
38
        this->gamesPlayed = 1;
39
        this->id = player.getId();
40 }
41
42 void PlayerStatWrap::loadStats(std::ifstream & file)
43
   {
44
        std::string line;
        std::getline(file, line, ',');
45
46
        this->name = line;
47
48
        std::getline(file, line, ',');
49
        this->id = std::stoi(line);
50
        std::getline(file, line, ',');
51
52
       this->wins = std::stoi(line);
53
54
        std::getline(file, line, ',');
55
        this->looses = std::stoi(line);
```

```
56
 57
         std::getline(file, line, ',');
 58
         this->timePlayed = std::stoi(line);
 59
         std::getline(file, line);
 60
 61
         this->gamesPlayed = std::stoi(line);
 62
 63 }
 64
 65 void PlayerStatWrap::saveStats(std::ofstream & file)
 66 {
        file << this->name << ","</pre>
 67
            << this->id << ","
 68
             << this->wins << ","
 69
             << this->looses << ","</pre>
 70
             << this->timePlayed << ","</pre>
 71
 72
             << this->gamesPlayed;
 73 }
 74 int PlayerStatWrap::get_timePlayed() const
 75 {
 76
         return this->timePlayed;
 77
    }
 78
 79 int PlayerStatWrap::get_gamesPlayed() const
 80 {
         return this->gamesPlayed;
 81
 82 }
 83
 84 int PlayerStatWrap::get_wins() const
 85 {
 86
         return this->wins;
 87 }
 88
 89 int PlayerStatWrap::get looses() const
 91
         return this->looses;
 92 }
 93
 94 int PlayerStatWrap::get_id() const
 95 {
 96
         return this->id;
 97 }
 98
 99 std::string PlayerStatWrap::get_name() const
100 {
101
         return this->name;
102 }
103
104 PlayerStatWrap & PlayerStatWrap::operator+(const PlayerStatWrap & src)
105 {
106
         this->wins += src.wins;
107
         this->looses += src.looses;
108
        this->timePlayed += src.timePlayed;
109
        this->gamesPlayed += src.gamesPlayed;
         return *this;
110
111 }
```

```
C:\risk_final\risk_final\PlayerStatWrap.cpp
```

```
113 PlayerStatWrap & PlayerStatWrap::operator=(const PlayerStatWrap & src)
        this->name = src.name;
        this->wins = src.wins;
        this->looses = src.looses;
        this->timePlayed = src.timePlayed;
        this->gamesPlayed = src.gamesPlayed;
        this->id = src.id;
```

112

114 { 115

116

117 118

119 120

121 122 } 123

return *this;

```
1 #pragma once
 2 #include <QWidget>
3 #include "ui_Mess.h"
 4 //#include "ui_Mess.h"
 6 class Mess : public QWidget {
 7
      Q_OBJECT
 8
 9 public:
      Mess(QWidget * parent = Q_NULLPTR);
10
11
       ~Mess();
      void setMess(std::string str);
12
13 private:
14
       Ui::mess_gui ui;
15
16
17 };
```

C:\risk_final\risk_final\Mess.cpp

```
1 #include "Mess.h"
 3 Mess::Mess(QWidget *parent) :QWidget(parent)
 4 {
 5
       ui.setupUi(this);
 6 }
 7 Mess::~Mess()
 8 {
10 }
11
12 void Mess::setMess(std::string str)
13 {
14
       this->ui.textEdit->setText(str.c_str());
15
       this->ui.textEdit->show();
16 }
17
```

C:\risk_final\risk_final\risk_final\statsGuiMain.h

```
1 #pragma once
 2 #include "ui_stats.h"
3 #include "PlayerStats.h"
 4 class statsGuiMain :public QWidget {
 5
       Q_OBJECT
 7 public:
     statsGuiMain(QWidget * parent = Q_NULLPTR);
 8
 9
      ~statsGuiMain();
10
11
      void setStats(PlayerStats* stats);
12
      void clear();
13 private:
       Ui::statsGui ui;
14
15
       PlayerStats* stats;
16 private slots:
17
      void spin(int);
18
       void player_stats(int);
19 };
```

C:\risk_final\risk_final\risk_final\statsGuiMain.cpp

```
1 #include "statsGuiMain.h"
 3 statsGuiMain::statsGuiMain(QWidget * parent) : QWidget(parent)
 4 {
 5
        ui.setupUi(this);
 6 }
 7
 8 statsGuiMain::~statsGuiMain()
 9 {
10
11 }
12
13 void statsGuiMain::setStats(PlayerStats * stats)
14 {
15
        this->stats = stats;
16
        QString text;
17
        text = (QString)this->stats->getAvgTime().c_str() + "\n";
18
        this->ui.all_stats->setText(text);
19
        const Vector<PlayerStatWrap> & players = this->stats->get_players();
        ui.chosenStats 2->insertItem(0, "");
21
        for (int i = 0; i<players.size(); i++)</pre>
22
23
            ui.chosenStats 2->insertItem(i+1, players[i].get name().c str());
24
        }
25 }
26
27 void statsGuiMain::clear()
28 {
29
        this->stats->clear();
30 }
31
32 void statsGuiMain::player stats(int index)
33 {
34
        index = index--;
35
        const Vector<PlayerStatWrap> & players = this->stats->get_players();
        const PlayerStatWrap & player = players[index];
36
        std::string mes="The player: "+ player.get_name()+ "\n wins: "
37
          +std::to_string(player.get_wins()) + " times \n loose: " +
38
            std::to_string(player.get_looses()) + " times \n played: "
              +std::to_string(player.get_gamesPlayed()) +
            " games \n for time: "+ std::to_string(player.get_timePlayed()/60) +"
39
             min":
40
        this->ui.Info_2->setText(mes.c_str());
41
        this->ui.Info 2->selectAll();
42
        this->ui.Info 2->setFontPointSize(12);
43
44 }
45
46 void statsGuiMain::spin(int maxSel)
47
48
        PlayerStats::Info<int> info;
        QString text = "name: ";
49
50
        QString afer_text = "";
51
        switch (maxSel) {
52
        case 0://wins
53
            info = this->stats->getMaxWins();
```

```
C:\risk_final\risk_final\risk_final\statsGuiMain.cpp
54
            text = QString(info.owner.c_str()) + " got max wins of: ";
55
            break;
56
       case 1://losses
57
            info = this->stats->getMaxLooses();
58
            text = QString(info.owner.c_str()) + " got max losses of: ";
59
           break;
       case 2://games
60
            info = this->stats->getMaxGamesPlayed();
61
            text = QString(info.owner.c_str()) + " got max games of: ";
62
63
           break;
64
       case 3://time
           info = this->stats->getMaxTimePlayed();
65
            info.value /= 60;
66
           text = QString(info.owner.c_str()) + " got max game time of: ";
67
            afer_text = QString(" min");
68
69
           break;
70
       }
71
72
73
       text = text + QString(std::to_string(info.value).c_str())+ afer_text;
74
       this->ui.Info->setText(text);
75 }
76
```

```
1 #pragma once
2 #include <QWidget>
3 #include "ui_move.h"
4 #include <iostream>
5 #include <QDebug>
 6 #include "Country.h"
7 class Move : public QWidget {
       Q_OBJECT
8
9
10 public:
11
      Move(QWidget * parent = Q_NULLPTR);
12
       ~Move();
13
      void setMax(int);
14
15
       void setMin(int);
16
       int getBonusTroops(int sel = 0);
17
       void setFrom(Country*);
18
       void setTo(Country*);
19 private:
20
       QWidget* father;
       Ui::Move ui;
21
22
       int min, max;
23
       Country* to;
       Country* from;
24
25 private slots:
      void plus(int sel = 0);
26
       void minus(int sel = 0);
27
28
29
       void end();
30 };
```

```
1 #include "Move.h"
 3 Move::Move(QWidget * parent) : QWidget(parent){
 4
       ui.setupUi(this);
 5
       this->min = 1;
 6
       this->max = 1;
 7
       this->to = nullptr;
 8
       this->from = nullptr;
 9
       this->ui.troopsP->setText("1");
       this->ui.ENTER->show();
10
       this->ui.MINUS->show();
11
12 }
13 Move::~Move()
14 {
15
16 }
17
18 void Move::setMax(int max)
19 {
20
       this->max = max;
21 }
22
23 void Move::setMin(int min)
24 {
25
       this->min = min;
       this->ui.troopsP->setText(std::to_string(min).c_str());
26
27 }
28
29 int Move::getBonusTroops(int sel)
30 {
31
       std::string text = this->ui.troopsP->toPlainText().toStdString();
32
       return std::stoi(text);
33 }
34
35 void Move::setFrom(Country * from)
36 {
37
       this->from = from;
38 }
39
40 void Move::setTo(Country * to)
41 {
42
       this->to = to;
43 }
44
45
46 void Move::end() {
47
       this->to->incTroops(this->getBonusTroops());
48
       this->from->incTroops((-1)*(this->getBonusTroops()));
49
       this->close();
50
       //this->father->setFocus();
51
52 }
54 void Move::plus(int sel)
55 {
56
       int number = this->getBonusTroops() + 1;
```

```
C:\risk_final\risk_final\Move.cpp
57
       if(number <= this->max)
58
           this->ui.troopsP->setText(std::to_string(number).c_str());
59 }
60
61 void Move::minus(int sel)
62 {
63
       int number = this->getBonusTroops() - 1;
64
       if (number >= this->min)
65
           this->ui.troopsP->setText(std::to_string(number).c_str());
66 }
```

67