Project title	**Betalfild**
Author(s)	**Bud Bogdan & Indre Bogdan**
Group	**30422**

->Classes realized by Bud Bogdan: Interface;

->Classes realized by Indre Bogdan: Division, Base, Battlefield;

1. Task Description

The game Betalfild is a turn-based strategy/luck game. It is basically an improved version of ROCK-PAPER-SCISSORS. It simulates the battle between two medieval armies commanded by two players. Once you press PLAY, you are prompted with a selection window for each player. During the selection process, each player must choose a name and buy troops (each player is provided with 1000 gold). There are four kinds of troops: Archers, Footmen, Cavalry and Trebuchets, each having a different price, a different HP and Attack and different weaknesses against other types of troops.
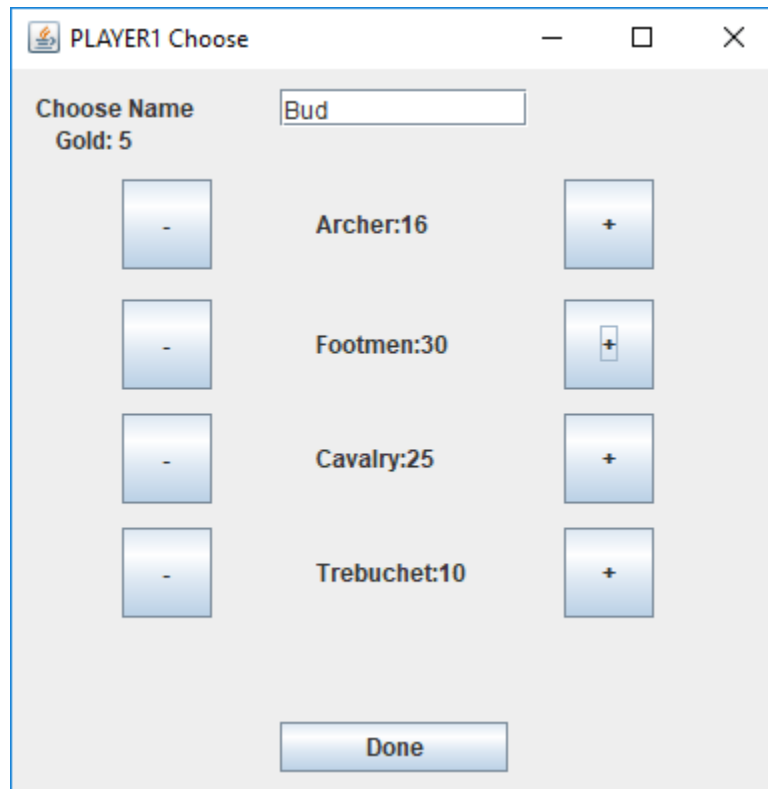
Once the selection process has ended, the fight begins. The fight is played in turns. The first phase of a turn consist in choosing how many troops each player wants to send on the four available battle tiles (at most one type of troop can occupy a tile). During this phase, players are required to look away when the opposing player sends his troops into battle (because the game is played on one screen and the element of surprise and chance is key to experiencing the game). There is one catch though. If a middle tile is left empty, and behind it, a base tile has unprotected troops, these troops can be attacked without being able to fight back. Once each player decides what troops he/she wants to send into battle (by pressing DONE) , the middle tiles are revealed (by pressing the REVEAL button). After the troops have been revealed, the total HP and attack of the combating troops are computed and tiles that are face to face start fighting (after the BATTLE button is pressed). This is when a turn ends. Any surviving troops return to their bases and the process repeats itself all over again.

The game ends, if, at the end of a turn a player is left with no troops in his/her base.
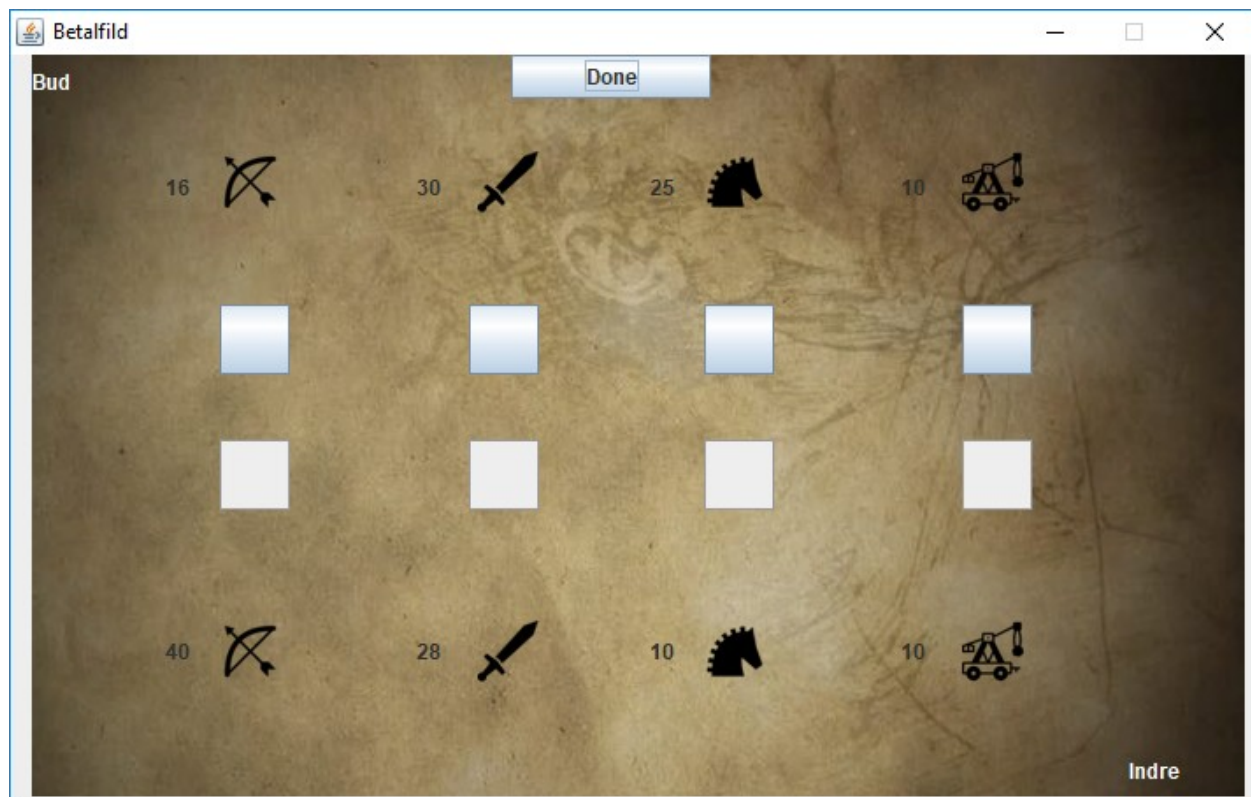
Initial Screen
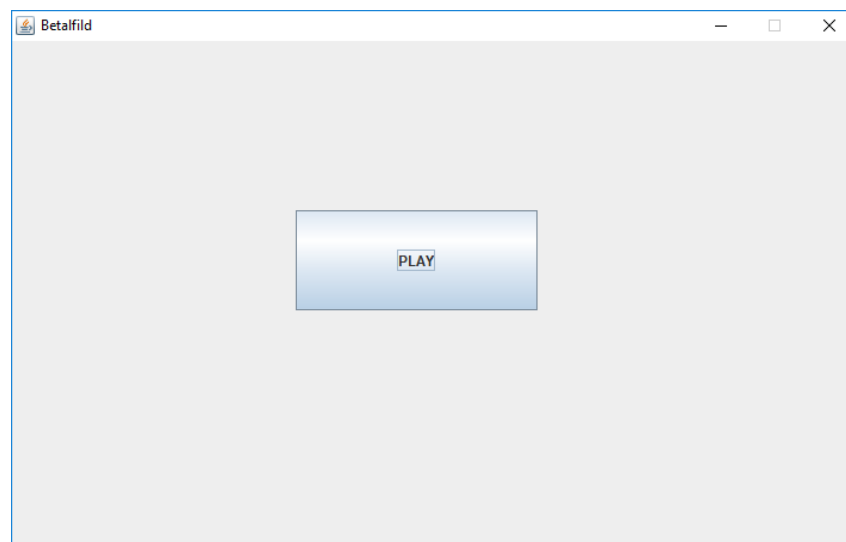
Player selection screen

Screen after buying soldiers

Select what units go into combat
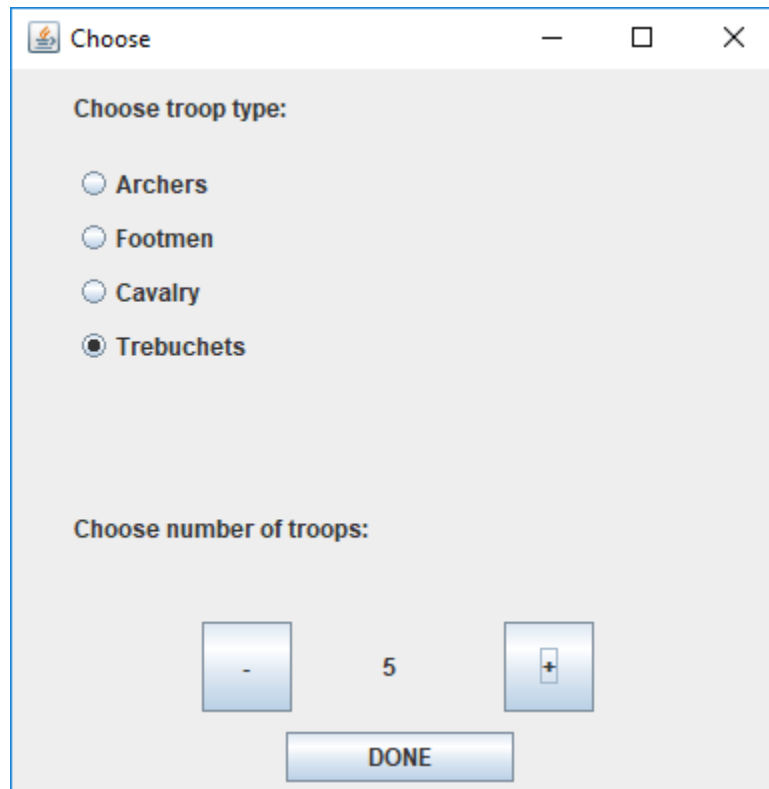
Troops after being revealed

Final screen. Showing the winner of the game
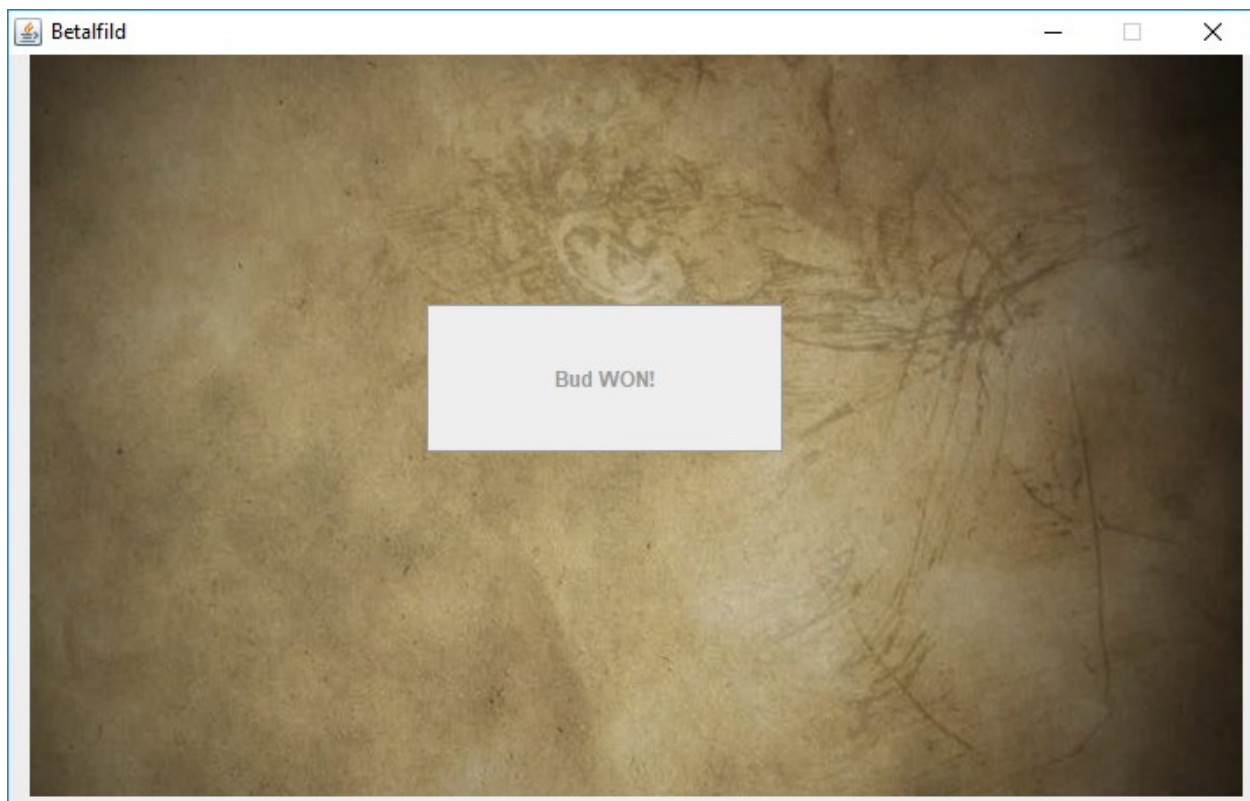
i. 2. Class Discovery

Base	
Buys/Sells soldiers	Division
Checks Soldiers	Division

Battlefield	
Send to battle	Base
	Division
Return from battle	Base
	Division
Attack	Base
	Division

3.Class Diagram

```
<<Java Class>>
   Ⓖ Base
    battle

○ STARTING_MONEY: int
◘ currentMoney: int
○ Name: String
◘ UNIT_PRICE: int[]

● Base(String)
● getCurrentMoney():int
● setCurrentMoney(int):void
● getName():String
● setName(String):void
● getDivisions(int):Division
● buySoldier(int):void
● sellSoldier(int):void
● checkSoldiers():void
● toString():String
```

```
<<Java Class>>
  Ⓖ Battlefield
    battle

● Battlefield()
● getPlayer(int):Base
● setPlayer(Base[]):void
● getPlace(int,int):Division
● setPlace(Division[][]):void
● sendToBattle(int,int,int,int):boolean
● attack(Division,Division,int):void
● battle():void
● returnFromBattle():void
● endGame():int
● toString():String
```

+player
0..*

-divisions
0..*

+place
0..*

```
<<Java Class>>
   Ⓖ Division
     battle

◘ type: int
◘ MAX_NR_OF_SOLDIERS: int
◘ currentNrOfSoldiers: int
◘ UNIT_HP: int
◘ totalHP: double
◘ UNIT_STRENGTH: int
◘ totalStrength: double
◘ dead: boolean

● Division(Division)
● Division(int,int)
● getType():int
● getUNIT_HP():int
● getCurrentNrOfSoldiers():int
● setCurrentNrOfSoldiers(int):void
● addSoldier():boolean
● killSoldier():boolean
● getTotalHP():double
● getTotalStrength():double
● isDead():boolean
● setDead(boolean):void
● reset():void
● toString():String
```