A C++ TEXT BASED GAME

KOKOVICS EMANUEL-ATTILA

Major: Informatică(engleză) - second year

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

WEST UNIVERSITY OF TIMIŞOARA

February 18^{th}, 2020

MAKING THE CARDS OF THE GAME

- each card is characterized by a number (or rank), a type (or suit);
- the other field are needed for the functionality of the game;
- the class contains standard get methods and a custom constructor;

PlayerCard	
-	cardNumber: char
-	cardType: char
-	isCardFaceUp: bool
+	parent: PlayerCard*
+	child: PlayerCard*
+	PlayerCard()
+	~PlayerCard()
+	PlayerCard(char, char)
+	getIsCardFaceUp(): bool
+	getCardNumber(): char
+	getCardType(): char
+	getKlondikeValue(): int
+	ostream& operator<< (std::ostream &, PlayerCard&)
+	Flip(): void

MAKING THE CARDS OF THE GAME

- the custom constructor:

```
PlayerCard::PlayerCard(char number, char type){
cardNumber = number;
cardType = type;
isCardFaceUp = false;
child = NULL;
parent = NULL;
}
```

- the Flip method:

```
void PlayerCard::Flip(){
isCardFaceUp = !isCardFaceUp;
}
```

MAKING THE CARDS OF THE GAME

```
int PlayerCard::getKlondikeValue(){
if(cardNumber == 'A')
return 1;
else if(cardNumber == 'T')
return 10;
else if(cardNumber == 'J')
return 11;
else if(cardNumber == 'Q')
return 12;
else if(cardNumber == 'K')
return 13;
else{
char card[] = {cardNumber, ''};
return atoi(card);
}
}
```

MAKING THE TABLES/COLUMNS OF THE GAME

- this class helps in the structure of the game;
- this class is used to create all the necessary columns of the game;

TableauPlayerCard	
-	actualMaximumAllowedSize: int
-	actualSize: int
-	playerCards: vector<PlayerCard*>
+	TableauPlayerCard()
+	~TableauPlayerCard()
+	TableauPlayerCard(int)
+	operator[](int): PlayerCard&
+	topPlayerCard(): PlayerCard&
+	pushCardOnStack(PlayerCard &): bool
+	pushValueOfCopyCard(PlayerCard card): bool
+	emptyTableau(): bool
+	removePlayerCardAtIndex(int index): bool
+	popLastCard(): bool
+	theSize: int
+	clearTableauOfPlayerCards(): void
+	moveBetween(TableauPlayerCard &, TableauPlayerCard &): static void

MAKING THE TABLES/COLUMNS OF THE GAME

- the custom constructor:

```
TableauPlayerCard::TableauPlayerCard(int maximumAllowedSize){
actualMaximumAllowedSize = maximumAllowedSize;
actualSize = 0;
playerCards.resize(maximumAllowedSize);
}
```

- method for adding on a card on a pile :

```
bool TableauPlayerCard::pushCardOnStack(PlayerCard
card){
if(actualSize < actualMaximumAllowedSize){
playerCards[actualSize] = &card;
actualSize++;
return true;
}
return false;
}
```

MAKING THE TABLES/COLUMNS OF THE GAME

- this method is to verify if there are any cards in the deck:

```
bool TableauPlayerCard::emptyTableau(){
if(actualSize == 0)
return true;
return false;
}
```

- this method returns card at last index:

```
PlayerCard& TableauPlayerCard::topPlayerCard(void){
if(actualSize > 0)
return *playerCards[actualSize-1];
}
```

MAKING THE GAME DECK

- this class makes use of the card class;
- this class will be used to make a standard 52 card deck;

MAKING THE GAME DECK

- the custom constructor:

```
StackDeckCards::StackDeckCards(){
currentIndex = 0;
srand((unsigned)time(0));
colonizeStackDeck();
}
```

- method that creates the deck:

```
void StackDeckCards::colonizeStackDeck(){
int index = 0;
for(int i=0; i<TYPE_CARD_SIZE; i++){
for(int j=0; j<NUMBER_CARD_SIZE; j++){
someRandomDeck[index] = PlayerCard(NUMBERS[j], TYPES[i]);
index++;
}
}
}
```

THE GAME SPECIFIC CLASS

- this class is specific to this game;
- it mainly contains methods that implement the rules, mathods that validate and make moves;
- it also contains a method that verifies that the game has ended;
- utilizes the class that creates the table/columns;