COMP4107 Feature Implementation Document

Group 1

Developer: YU Fengfei-21251215

In this group project, I'm in charged of the following codes.

1. WeaponCard Factory

Explanation:

This is a Singleton WeaponCard Factory, called in `cardGenerator` from General Manager to create weapon cards.

In this factory, there is only one method called `createCard` overriding the super `createCard` method to return `WeaponCard`s.

In this method, a var `weaponCardList` holds all the possible weapon cards with their suit and rank (e.g., `Zhuge Crossbow,Spade,A`).

A piece of information `random` is randomly selected and split by from the `weaponCardList`. A val `card` is randomly selected from the list. A val `suits` is a String list holds four suits extracted from `random[1]`. A val `ranks` is a String list holds thirteen ranks extracted from `random[2]`.

In each call of `WeaponCardFactory.createCard()`, a weapon `card` will be created and assigned with proper suit and rank.

Finally, it returns a card to `cardGenerator()`.

Card Factory Code Snippet:

```
abstract class CardFactory {
    abstract fun createCard(): Card

}
```

WeaponCard Factory Code Snippet:

```
class WeaponFactory: CardFactory(){
    var weaponCardList: MutableList<String> = mutableListOf(
        "Zhuge Crossbow,Spade,A", "Zhuge Crossbow,Diamond,A",
        "Yin-Yang Swords,Club,2", "Yin-Yang Swords,Heart,K",
        "Green Dragon Blade,Club,5",
        "Serpent Spear,Club,Q",
        "Rock Cleaving Axe,Diamond,5",
```

```
            "Kirin Bow,Heart,5",
            "Blue Steel Blade,Club,6"
        )
        override fun createCard(): Card {
            var random = weaponCardList.random()
            weaponCardList.remove(random)
            var information = random.split(",")
            val type = information[0]
            val suit = information[1]
            val rank = information[2]
            var card = when (type) {
                "Zhuge Crossbow" -> Zhuge_Crossbow(null,null)
                "Yin-Yang Swords" -> Yin_Yang_Swords(null,null)
                "Green Dragon Blade" -> Green_Dragon_Blade(null, null)
                "Serpent Spear" -> Serpent_Spear(null,null)
                "Rock Cleaving Axe" -> Rock_Cleaving_Axe(null,null)
                "Kirin Bow" -> Kirin_Bow(null,null)
                "Blue Steel Blade" -> Blue_Steel_Blade(null,null)
                else -> throw IllegalArgumentException("Invalid Weapon Card")
            }
            card.suit = suit
            card.rank = rank
            weaponCardList.remove(random)
            return card
        }
    }
```

Sample Output:

```
♦ 5 Rock Cleaving Axe, ♡ 5 Kirin Bow
```

2. attack() in general Strategy / ZhaoYunStrategy / LiuBeiStrategy

Explanation:

In Strategy.kt, I revise the attack() function to simulate real gaming situations with `weapon`. For each weapon, it can be an `active weapon (Yin-Yang Swords)` or `passive weapon (e.g., Zhuge Crossbow)`.

I divide the attacking phase into three partitions, namely `Before attacking`, `During attacking` and `After attacking`since some weapons' skills need to be executed at different stages of attack.

General attack() Code Snippet:

```
override fun attack() {
        if (general.hasAttackCard()) {

            var tar: General? = null
```

```
            for (gen in generals) {
                if (gen.player is Lord) {
                    tar = gen
                }
            }
//      ********************************************* SET ATTACK TIMES
            var attackingNum = 0                                                    //
default
            if (general.weapon is Zhuge_Crossbow) {                                 //
Zhuge Crossbow
                (general.weapon as Zhuge_Crossbow).execute(general, null)
                attackingNum = general.hand.size                                    //
Zhuge Crossbow: maximum number of attacks = cards in hand
            }
            else if (general.weapon is Serpent_Spear) {
// Serpent Spear: increase 1 attack time by discarding 2 cards
                (general.weapon as Serpent_Spear).execute(general, tar)
                attackingNum++
            } else {
                attackingNum = 1
            }
            if (general is ZhangFei) {
                println("[Berserk] Zhang Fei can use as many ATTACK cards as he
wishes during the turn.")
                attackingNum = general.hand.size                                    //
Zhang Fei's Skill: maximum number of attacks = cards in hand
            }
//      ********************************************* SET ATTACK TIMES

            while (attackingNum > 0) {
                var attackCard = Attack(null, null)
                var can = false
                for (card in general.hand) {
                    if (card is Attack) {
                        checkRange(general)

                        if (general.attackRange >= checkDistance(general, tar!!))
{
                            attackCard = card
                            attackCard.receiver = tar
                            can = true
                            println("${general.name} can attack target general
${tar.name}.")
                            break
                        }
                    }
                }
                if (can) {
//      ******************************************** BEFORE ATTACK
                    if (general.weapon is Yin_Yang_Swords) {
// Yin-Yang Swords
                        (general.weapon as Yin_Yang_Swords).execute(general, tar)
                    } else if (general.weapon is Blue_Steel_Blade) {
// Blue Steel Blade
```

```
                        (general.weapon as Blue_Steel_Blade).execute(general, tar)
                    }
//      ****************************************** BEFORE ATTACK


//      ------------------------------------------ DURING ATTACK
                    println(
                        "${general.name} spends a card
${general.suitMap[attackCard.suit]} " +
                                "${attackCard.rank} ${attackCard.name} to attack
${tar!!.name}."
                    )
                    val originalTarHP = tar.currentHP
                    tar.beingAttacked(general, attackCard.suit!!)
//      ------------------------------------------ DURING ATTACK


//      ############################################## AFTER ATTACK
                    if (tar.defense != null) {
                        tar.defense!!.valid =
                            true                                    // Blue Steel
Blade: reset the defense card
                    }

                    val currentTarHP = tar.currentHP

                    if (originalTarHP > currentTarHP) {
// Kirin Bow
                        if (general.weapon is Kirin_Bow) {
                            (general.weapon as Kirin_Bow).execute(general, tar)
                        }
                    } else {
                        if (general.weapon is Rock_Cleaving_Axe) {
// Rock Cleaving Axe
                            (general.weapon as Rock_Cleaving_Axe).execute(general,
tar)
                        } else if (general.weapon is Green_Dragon_Blade) {
// Green Dragon Blade
                            (general.weapon as
Green_Dragon_Blade).execute(general, null)
                            attackingNum++
                        }
                    }
                    general.hand.remove(attackCard)
                    discardDeck.add(attackCard)
                    attackingNum--                                          // update
attack times
//      ############################################## AFTER ATTACK
                    if (tar is CaoCao) {
                        var helpCao = false
                        if (tar.nextGeneral != null) {
                            tar.nextGeneral!!.canHelpLord(general)
                            tar.canActivateEntourageList.forEach {
                                if (it) {
```

```
                                    helpCao = true
                                }
                            }
                        }
                        if (!helpCao) {
                            tar.treachery(attackCard)
                        }
                    }
                } else {
                    if (general.attackRange < checkDistance(general, tar!!)) {
                        print("(Distance) ")
                    }
                    println("${general.name} cannot attack target general
${tar!!.name}.")
                    break
                }
            }

        } else if (!general.hasAttackCard() && (general.weapon is Serpent_Spear)
&& general.hand.size >= 4) {
            val tar = rebelList.random()
            (general.weapon as Serpent_Spear).execute(general, tar)
        } else {
            println("${general.name} doesn't have an attack card.")
        }

    }
```

Zhao Yun attack() Code Snippet:

```
class ZhaoYunStrategy(val general: General) : Strategy(general) {
    open lateinit var identityStrategy: Strategy
    override fun attack() {
        var tar: General? = null
        if (general.player is Loyalist || (general.player is Spy &&
!general.player.isRevealed)) {
            if (rebelList.size == 0) {
                return
            }
            tar = rebelList.random()
        } else {
            tar = lord
        }
        if (general.hasAttackCard()) {
//      ****************************************** SET ATTACK TIMES
            var attackingNum = 0                                                //
default
            if (general.weapon is Zhuge_Crossbow) {                             //
Zhuge Crossbow
                (general.weapon as Zhuge_Crossbow).execute(general, null)
                attackingNum =
```

```
                    general.hand.size                                   // Zhuge
Crossbow: maximum number of attacks = cards in hand
            } else {
                attackingNum = 1
            }
//      ****************************************** SET ATTACK TIMES

            while (attackingNum > 0) {
                var attackCard: Card? = null
                var can = false
                for (card in general.hand) {
                    if (card is Attack) {
                        checkRange(general)

                        if (general.attackRange >= checkDistance(general, tar!!))
{
                            attackCard = card
                            attackCard.receiver = tar
                            can = true
                            println("Zhao Yun can attack general ${tar.name}.")
                            break
                        }
                    }
                    if (card is Dodge) {
                        checkRange(general)

                        if (general.attackRange >= checkDistance(general, tar!!))
{
                            attackCard = card
                            attackCard.receiver = tar
                            can = true
                            println("Zhao Yun can attack general ${tar.name}.")
                            break
                        }
                    }
                }
                if (can) {
//      ****************************************** BEFORE ATTACK
                    if (general.weapon is Yin_Yang_Swords) {
// Yin-Yang Swords
                        (general.weapon as Yin_Yang_Swords).execute(general, tar)
                    } else if (general.weapon is Blue_Steel_Blade) {
// Blue Steel Blade
                        (general.weapon as Blue_Steel_Blade).execute(general, tar)
                    }
//      ****************************************** BEFORE ATTACK


//      ------------------------------------------ DURING ATTACK
                    println(
                        "${general.name} spends a card
${general.suitMap[attackCard!!.suit]} " +
                                "${attackCard!!.rank} ${attackCard!!.name} to
attack ${tar!!.name}."
```

```
                    )
                    val originalTarHP = tar.currentHP
                    tar.beingAttacked(general, attackCard.suit!!)
//      ------------------------------------------- DURING ATTACK


//      ############################################# AFTER ATTACK
                    if (tar.defense != null) {
                        tar.defense!!.valid =
                            true                                    // Blue Steel
Blade: reset the defense card
                    }

                    val currentTarHP = tar.currentHP

                    if (originalTarHP > currentTarHP) {
// Kirin Bow
                        if (general.weapon is Kirin_Bow) {
                            (general.weapon as Kirin_Bow).execute(general, tar)
                        }
                    } else {
                        if (general.weapon is Rock_Cleaving_Axe) {
// Rock Cleaving Axe
                            if(tar.currentHP > 0)
                                (general.weapon as
Rock_Cleaving_Axe).execute(general, tar)
                        } else if (general.weapon is Green_Dragon_Blade) {
// Green Dragon Blade
                            (general.weapon as
Green_Dragon_Blade).execute(general, null)
                            attackingNum++
                        }
                    }
                    general.hand.remove(attackCard)
                    discardDeck.add(attackCard)
                    attackingNum--                                          // update
attack times
//      ############################################# AFTER ATTACK
                    if (tar is CaoCao && lord is CaoCao) {
                        var helpCao = false
                        if (tar.nextGeneral != null) {
                            tar.nextGeneral!!.canHelpLord(general)
                            tar.canActivateEntourageList.forEach {
                                if (it) {
                                    helpCao = true
                                }
                            }
                        }
                        if (!helpCao) {
                            tar.treachery(attackCard)
                        }
                    }
                    if(tar.currentHP <= 0){
                        break
```

```
                    }
                } else {
                    if (general.attackRange < checkDistance(general, tar!!)) {
                        print("(Distance) ")
                    }
                    println("Zhao Yun cannot attack general ${tar.name}.")
                    break
                }
            }

        } else {
            if ((general.weapon is Serpent_Spear) && general.hand.size >= 4) {
                val tar = rebelList.random()
                (general.weapon as Serpent_Spear).execute(general, tar)
            } else {
                println("${general.name} doesn't have an attack card.")
            }
        }
    }

  ...The rest code are ignored
}
```

Liu Bei Strategy Code Snippet:

```
class LiuBeiStrategy(general: General) : LoyalistStrategy(general) {
    lateinit var state: State

    override fun playNextCard() {
        checkPlayers()
        spyReveal()
        checkPlayers()
        state.playNextCard()
        super.horse()
        super.weapon()
        super.defense()

        super.recovery()
        super.useCommandCard()
        attack()
        commandCards.clear()
        loyalistList.clear()
        rebelList.clear()
    }

    open fun initializeStrategy() {
        if (general.currentHP >= 2) {
            state = HealthyState(general.strategy as LiuBeiStrategy)
        } else {
            state = UnhealthyState(general.strategy as LiuBeiStrategy)
        }
```

```
    }

    override fun attack() {
        var tar: General? = null
        if (rebelList.size == 0) {
            return
        }
        tar = rebelList.random()

        if (general.hasAttackCard()) {
//      ****************************************** SET ATTACK TIMES
            var attackingNum = 0                                                    //
default
            if (general.weapon is Zhuge_Crossbow) {                                 //
Zhuge Crossbow
                (general.weapon as Zhuge_Crossbow).execute(general, null)
                attackingNum =
                    general.hand.size                                   // Zhuge
Crossbow: maximum number of attacks = cards in hand
            } else {
                attackingNum = 1
            }
//      ****************************************** SET ATTACK TIMES

            while (attackingNum > 0) {
                var attackCard: Card? = null
                var can = false
                for (card in general.hand) {
                    if (card is Attack) {
                        checkRange(general)

                        if (general.attackRange >= checkDistance(general, tar!!))
{
                            attackCard = card
                            attackCard.receiver = tar
                            can = true
                            println("Liu Bei can attack general ${tar.name}.")
                            break
                        }
                    }
                }
                if (can) {
//      ****************************************** BEFORE ATTACK
                    if (general.weapon is Yin_Yang_Swords) {
// Yin-Yang Swords
                        (general.weapon as Yin_Yang_Swords).execute(general, tar)
                    } else if (general.weapon is Blue_Steel_Blade) {
// Blue Steel Blade
                        (general.weapon as Blue_Steel_Blade).execute(general, tar)
                    }
//      ****************************************** BEFORE ATTACK


                    val rouse = (general as LiuBei).RouseJudgement()
```

```
//      ------------------------------------------- DURING ATTACK
                    if(rouse == false){
                        println(
                            "${general.name} spends a card
${general.suitMap[attackCard!!.suit]} " +
                                    "${attackCard.rank} ${attackCard.name} to
attack ${tar.name}."
                        )

                        val originalTarHP = tar.currentHP
                        tar.beingAttacked(general, attackCard.suit!!)
//      ------------------------------------------- DURING ATTACK

//      ############################################ AFTER ATTACK
                        if (tar.defense != null) {
                            tar.defense!!.valid =
                                true                                    // Blue
Steel Blade: reset the defense card
                        }

                        val currentTarHP = tar.currentHP

                        if (originalTarHP > currentTarHP) {
// Kirin Bow
                            if (general.weapon is Kirin_Bow) {
                                (general.weapon as Kirin_Bow).execute(general,
tar)
                            }
                        } else {
                            if (general.weapon is Rock_Cleaving_Axe) {
// Rock Cleaving Axe
                                if(tar.currentHP > 0)
                                    (general.weapon as
Rock_Cleaving_Axe).execute(general, tar)
                            } else if (general.weapon is Green_Dragon_Blade) {
// Green Dragon Blade
                                (general.weapon as
Green_Dragon_Blade).execute(general, null)
                                attackingNum++
                            }
                        }
                        general.hand.remove(attackCard)
                        discardDeck.add(attackCard)
                        attackingNum--                                      //
update attack times
//      ############################################ AFTER ATTACK
                        if (tar is CaoCao && lord is CaoCao) {
                            var helpCao = false
                            if (tar.nextGeneral != null) {
                                tar.nextGeneral!!.canHelpLord(general)
                                tar.canActivateEntourageList.forEach {
                                    if (it) {
                                        helpCao = true
                                    }
```

```
                                }
                            }
                            if (!helpCao) {
                                tar.treachery(attackCard)
                            }
                        }
                    }
                    else{
                        attackCard = (general as LiuBei).RouseAttack()
                        println(
                            "${general.name} spends a card
${general.suitMap[attackCard!!.suit]} " +
                                    "${attackCard.rank} ${attackCard.name} to
attack ${tar.name}."
                        )

//      ------------------------------------------- DURING ATTACK
                        val originalTarHP = tar.currentHP
                        tar.beingAttacked(general, attackCard.suit!!)

//      ############################################# AFTER ATTACK
                        val currentTarHP = tar.currentHP

                        if (tar.defense != null) {
                            tar.defense!!.valid =
                                true                                    // Blue
Steel Blade: reset the defense card
                        }

                        if (originalTarHP > currentTarHP) {
// Kirin Bow
                            if (general.weapon is Kirin_Bow) {
                                (general.weapon as Kirin_Bow).execute(general,
tar)
                            }
                        } else {
                            if (general.weapon is Rock_Cleaving_Axe) {
// Rock Cleaving Axe
                                if(tar.currentHP > 0)
                                    (general.weapon as
Rock_Cleaving_Axe).execute(general, tar)
                            } else if (general.weapon is Green_Dragon_Blade) {
// Green Dragon Blade
                                (general.weapon as
Green_Dragon_Blade).execute(general, null)
                                attackingNum++
                            }
                        }

                        attackingNum--                                      //
update attack times
//      ############################################# AFTER ATTACK
                        if (tar is CaoCao && lord is CaoCao) {
                            var helpCao = false
```

```
                            if (tar.nextGeneral != null) {
                                tar.nextGeneral!!.canHelpLord(general)
                                tar.canActivateEntourageList.forEach {
                                    if (it) {
                                        helpCao = true
                                    }
                                }
                            }
                            if (!helpCao) {
                                tar.treachery(attackCard)
                            }
                        }
                    }

                    if(tar.currentHP <= 0){
                        break
                    }
                } else {
                    if (general.attackRange < checkDistance(general, tar)) {
                        print("(Distance) ")
                    }
                    println("Liu Bei cannot attack general ${tar.name}.")
                    break
                }
            }

        } else {
            if ((general.weapon is Serpent_Spear) && general.hand.size >= 4) {
                val tar = rebelList.random()
                (general.weapon as Serpent_Spear).execute(general, tar)
            } else {
                println("${general.name} doesn't have an attack card.")
            }
        }
    }
}
```

3. WeaponCards

Explanation:

In WeaponCards.kt, there are totally seven kinds of weapons, which are `Zhuge Crossbow`, `Yin-Yang Swords`, `Green Dragon Blade`, `Serpent Spear`, `Rock Cleaving Axe`, `Kirin Bow` and `Blue Steel Blade`. They are inherited from WeaponCard class.

In the classes, val name are overridden into corresponded card name.

For each kind of weapon, a `distance` attribute will be assigned for future `checkDistance()` use.

Their corresponding functionalities are shown below.

---

1. Zhuge Crossbow:

It is a passive weapon (i.e., when used, only its existence is checked to judge whether its owner has the weapon skill).

This weapon can enable its owner to spend as many `Attack` as he/she wishes.

Weapon Code Snippet:

```
class Zhuge_Crossbow(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver) {
    override val name = "Zhuge Crossbow"
    override var distance = 1

    override fun execute(initializer: General, receiver: General?) {
        println("[Zhuge Crossbow] Continuous \"Attack\"s enabled.")
    }
}
```

Sample Output:

```
Liu bei draw 2 card(s), now has 6 card(s). Hand: ♡ 8 Peach, ♡ 9 Peach, ♡ 4
Peach, ♦ 5 Dodge, ♠ 10 Attack, ♣ J Attack
[Zhuge Crossbow] Continuous "Attack"s enabled.
Liu bei can attack target general Xu Chu.
Liu bei spends a card ♠ 10 Attack to attack Xu Chu.
Xu Chu being attacked.
Xu Chu is attacked successfully, current HP is 1
Liu bei can attack target general Xu Chu.
Liu bei spends a card ♣ J Attack to attack Xu Chu.
Xu Chu being attacked.
Xu Chu is attacked successfully, current HP is 0
```

---

2. Yin-Yang Swords:

It is an active weapon (i.e., when used, owner can choose to use or not use the weapon skill). In our game, the weapon skill is always used because it is likely to help its owner to win this game.

When the owner attacks a general with a different gender, the owner can choose to (`draw(1)`) one card or let the receiver to (`discard(1)`) one card.

Weapon Code Snippet:

```
class Yin_Yang_Swords(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver) {
    override val name = "Yin-Yang Swords"
```

```
    override var distance = 2
    override fun execute(initializer: General, receiver: General?) {
        if (receiver != null) {
            if(!initializer.gender.equals(receiver.gender)){
                val random = (0..1).random()
                if (random <= 0.5) {
                    receiver.discard(1)
                    println("[Yin-Yang Swords] ${initializer.name} chooses to let
${receiver.name} to discard one card.")
                } else {
                    initializer.draw(1)
                    println("[Yin-Yang Swords] ${initializer.name} chooses to let
himself/herself to draw one card.")
                }
            }
        }
    }
}
```

Sample Output:

```
// same gender
Sun Quan takes the weapon: [Yin-Yang Swords] from his/her hand to the weapon area.
Sun Quan can attack target general Xu Chu.
[Yin-Yang Swords] fails to take effect.

// different gender
Hua tuo takes the weapon: [Yin-Yang Swords] from his/her hand to the weapon area.
Hua tuo can attack target general Sun Shang Xiang.
[Yin-Yang Swords] Hua tuo chooses to let himself/herself to draw one card.
Hua tuo draw 1 card(s), now has 5 card(s). Hand: ♡ 6 Peach, ♦ 8 Dodge, ♦ Q
Peach, ♣ 4 Attack, ♡ A RainArrows
```

---

3. Green Dragon Blade:

It is a passive weapon (i.e., when used, only its existence is checked to judge whether its owner has the weapon skill).

In our game, the weapon skill is always used because it is likely to help its owner to win this game.

When the owner's attack is successfully dodged by the receiver, this weapon enables the owner to attack again until:

- The owner runs of out `Attack` cards.
- The receiver is successfully attacked.

Weapon Code Snippet:

```
class Green_Dragon_Blade(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver){
    override val name = "Green Dragon Blade"
    override var distance = 3
    override fun execute(initializer: General, receiver: General?) {
        println("[Green Dragon Blade] \"Attack\" enabled again.")
    }
}
```

Sample Output:

```
Hua Xiong takes the weapon: [Green Dragon Blade] from his/her hand to the weapon
area.
Hua Xiong can attack target general Zhen Ji.
Hua Xiong spends a card ♡ 10 Attack to attack Zhen Ji.
Zhen Ji being attacked.
[Eight Trigrams Formation]: used but with unsuccessful "Dodge".
Zhen Ji is attacked successfully, current HP is 0
Zhen Ji's hp is 0 now!
[Green Dragon Blade] "Attack" enabled again.
// Zhen Ji is dead, so Hua Xiong cannot attack again. But in this case, Hua Xiong
has a second Attack chance.
Hua Xiong cannot attack target general Zhen Ji.
```

---

4. Serpent Spear:

It is an active weapon (i.e., when used, owner can choose to use or not use the weapon skill). In our game, the weapon skill is always used because it is likely to help its owner to win this game.

When the general needs or uses an `Attack`, he/she can use two non-Attack cards to be a `pseudo-Attack`. Normally, it is at the last second will a general to sacrifice two cards to be an `Attack`.

Specifically, `when the general does not have Attack card but with sufficient non-Attack cards (i.e., card number >= 4)`, this general is willing to discard two cards to initialize an Attack `once`.

Moreover, it is necessary to spend an Attack against `Barbarian` since it could cause HP reduction. Therefore, we let the owner use Serpent Spear when faced with `Barbarian`.

Weapon Code Snippet:

```
class Serpent_Spear(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver) {
    override val name = "Serpent Spear"
    override var distance = 3
    override fun execute(initializer: General, receiver: General?) {
        if (initializer.hand.size >= 2) {
```

```
            initializer.discard(2)
            val list = mutableListOf("Heart", "Spade", "Diamond", "Club")
            if (receiver != null) {
                println("[Serpent Spear] ${initializer.name} spends an Attack to
${receiver.name} by discarding 2 cards.")
                receiver.beingAttacked(initializer, list.random())
            } else {
                println("[Serpent Spear] ${initializer.name} spends an Attack by
discarding 2 cards.")
            }
        }
    }
}
```

`Faced with Barbarian` **Code Snippet (partial code from CommandCards.kt):**

```
    if (gen.hasAttackCard() && !gen.equals(initializer)) {
        isDodged = true
        //spend one attack card to dodge the attack
        gen.spendAttackCard()
        println("${gen.name} successfully dodge the Barbarian")
    }
    if (gen.equals(initializer)) {
        isDodged = true
    }
    if(!gen.hasAttackCard() && !gen.equals(initializer) && gen.hand.size >= 2 &&
gen.weapon is Serpent_Spear){
        (gen.weapon as Serpent_Spear).execute(gen,null)
        println("${gen.name} successfully dodge the Barbarian")
    }
```

`Sufficient Cards to initialize an Attack` **Code Snippet (partial code from Strategy.kt):**

```
else if (!general.hasAttackCard() && (general.weapon is Serpent_Spear) &&
general.hand.size >= 4) {
  val tar = rebelList.random()
  (general.weapon as Serpent_Spear).execute(general, tar)
```

Sample Output:

```
// Xu Chu has no Attack card

Xu Chu draw 2 card(s), now has 5 card(s). Hand: ♦ 5 Dodge, ♦ 7 Dodge, ♦ 6 Dodge, ♣
9 Serpent Spear, ♠ 3 StealingSheep
Xu Chu takes the weapon: [Serpent Spear] from his/her hand to the weapon area.
Xu Chu steal a card Peach from Sun Shang Xiang. Hand: ♦ 5 Dodge, ♦ 7 Dodge, ♦ 6
```

```
Dodge, ♠ 3 StealingSheep, ♡ 8 Peach
Xu Chu discarded 2 card(s), now has 2 card(s). Hand: ♦ 5 Dodge, ♦ 7 Dodge
[Serpent Spear] Xu Chu spends an Attack to Sun Shang Xiang by discarding 2 cards.
Sun Shang Xiang being attacked.
Sun Shang Xiang's hp is 0 now!
Xu Chu kills Sun Shang Xiang, a Rebel, rewards 3 additional cards!


// Weapon fail to take effect

Liu bei draw 2 card(s), now has 6 card(s). Hand: ♠ 8 Attack, ♠ 10 Attack, ♡ 10
Attack, ♠ 3 StealingSheep, ♡ 8 Peach, ♣ 8 Serpent Spear
Liu bei takes the weapon: [Serpent Spear] from his/her hand to the weapon area.
Liu bei steal a card Dismantle from Lv Meng. Hand: ♠ 8 Attack, ♠ 10 Attack, ♡ 10
Attack, ♠ 3 StealingSheep, ♡ 8 Peach, ♣ K Dismantle
Liu bei can attack target general Lv Meng.
Liu bei spends a card ♠ 8 Attack to attack Lv Meng.
Lv Meng being attacked.
Lv Meng is attacked successfully, current HP is 3
// No following attack
Lv Meng draw 2 card(s), now has 5 card(s). Hand: ♣ 2 Attack, ♦ 9 Attack, ♡ 6
Peach, ♦ 8 Attack, ♡ 4 Peach


// Used against Barbarian

Liu bei launch a command Barbarian, every one need to spend an attack card to
avoid the attack
Xu Chu discarded 2 card(s), now has 0 card(s). Hand:
[Serpent Spear] Xu Chu spends an Attack by discarding 2 cards.
Xu Chu successfully dodge the Barbarian
```

---

5. Rock Cleaving Axe:

It is an active weapon (i.e., when used, owner can choose to use or not use the weapon skill). In our game, the weapon skill is always used because it is likely to help its owner to win this game.

When the target general successfully dodges the attack from the weapon owner, the owner could `force to deduct 1 HP from the reciver by discarding 2 cards.`

For this forcing effect, I create a special function `beingAttacked_Rock_Cleaving_Axe` to handle such attack.

Weapon Code Snippet:

```
class Rock_Cleaving_Axe(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver){
    override val name = "Rock Cleaving Axe"
    override var distance = 3
    override fun execute(initializer: General, receiver: General?) {
        if(initializer.hand.size >= 2 ){
```

```
            initializer.discard(2)
            val list = mutableListOf("Heart", "Spade", "Diamond", "Club")
            println("[Rock Cleaving Axe] \"Attack\" is forced to take effect by
${initializer.name} discarding 2 cards.")
            receiver!!.beingAttacked_Rock_Cleaving_Axe(initializer, list.random())
        }
    }
}
```

`Special beingAttacked handling` **Code Snippet (partial code from General.kt):**

```
open fun beingAttacked_Rock_Cleaving_Axe(sender: General, suit: String) {
    val before = currentHP
    currentHP--
    val after = currentHP
    println("$name being attacked by Rock Cleaving Axe. HP becomes from ${before}
to ${after}.")
    if (currentHP <= 0) {
        alive = false
        killer = sender
        currentHP = 0
        println("$name's hp is $currentHP now!")
    }
}
```

`Weapon Usage in Strategy` **Code Snippet (partial code from Strategy.kt):**

```
if (general.weapon is Rock_Cleaving_Axe) {                    // Rock Cleaving
Axe
    (general.weapon as Rock_Cleaving_Axe).execute(general, tar)
} else if (general.weapon is Green_Dragon_Blade) {            // Green Dragon
Blade
    (general.weapon as Green_Dragon_Blade).execute(general, null)
    attackingNum++
}
```

Sample Output:

```
Sun Quan takes the weapon: [Rock Cleaving Axe] from his/her hand to the weapon
area.
Sun Quan can attack target general Hua Xiong.
Sun Quan spends a card ♡ 10 Attack to attack Hua Xiong.
Hua Xiong being attacked.
[Eight Trigrams Formation] cannot help Hua Xiong dodge.
Hua Xiong dodge 1 attack by spending a DODGE card. Current cards: 3.
[Rock Cleaving Axe] "Attack" is forced to take effect by Sun Quan discarding 2
```

```
cards.
Hua Xiong being attacked by Rock Cleaving Axe. HP becomes from 6 to 5.
```

---

6. Kirin Bow:

It is a defined passive weapon. In our game, the weapon skill is always used because it is likely to help its owner to win this game.

`If the owner attacks a target general successfully, he/she could abandon a horse card.` Therefore, I introduce two variables `originalTarHP` and `currentTarHP` to test whether the Attack takes effect.

If there is no horse in the target general's `horse area`, this weapon fails to make an effect.

If the target general has two horses (i.e., `equipment` has two horseCard), we randomly abandon one of the horseCard.

Weapon Code Snippet:

```
class Kirin_Bow(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver) {
    override val name = "Kirin Bow"
    override var distance = 5
    override fun execute(initializer: General, receiver: General?) {
        if (receiver!!.equipment.size != 0) {
            val random = receiver.equipment.random()
            receiver.equipment.remove(random)
            discardDeck.add(random)
            println("[Kirin Bow] ${receiver.name}'s Horse Card: ${random.name} is
abandoned.")
        }
        else{
            println("[Kirin Bow] ${receiver.name} does not have a horse.")
        }
    }
}
```

`Weapon Usage in Strategy` **Code Snippet (partial code from Strategy.kt):**

```
val originalTarHP = tar.currentHP
tar.beingAttacked(general, attackCard.suit!!)
val currentTarHP = tar.currentHP
if (originalTarHP > currentTarHP) {
    if (general.weapon is Kirin_Bow) {
        (general.weapon as Kirin_Bow).execute(general, tar)
    }
}
```

Sample Output:

```
// No horse

Zhao Yun can attack target general Sun Quan.
Zhao Yun spends a card ♠ 8 Attack to attack Sun Quan.
Sun Quan being attacked.
Sun Quan is attacked successfully, current HP is 1
Current risk level: 4
[Kirin Bow] Sun Quan does not have a horse.


// Have horse

Liu bei can attack target general Lv Bu.
Liu bei spends a card ♡ J Attack to attack Lv Bu.
Lv Bu being attacked.
Lv Bu is attacked successfully, current HP is 3
[Kirin Bow] Lv Bu's Horse Card: Shadowrunner is abandoned.
```

---

7. Blue Steel Blade:

It is a passive weapon. In our game, the weapon skill is always used because it is likely to help its owner to win this game.

`This weapon ignore the effect of defenseCard.` Thus, I introduce a boolean variable `valid` in `defenseCard` to clarify whether the defense card can be used.

In the attack() strategy in `Strategy.kt`, if the general's weapon is Blue Steel Blade and the receiver has a defense card, the `valid` is set to false. After this attack, the `valid` is set back to true.

Weapon Code Snippet:

```
class Blue_Steel_Blade(initializer: General?, receiver: General?) :
WeaponCard(initializer, receiver) {
    override val name = "Blue Steel Blade"
    override var distance = 2
    override fun execute(initializer: General, receiver: General?) {
        if (receiver!!.defense != null) {
            receiver.defense!!.valid = false
            println("[Blue Steel Blade] ${receiver.name}'s Defense Card:
${receiver.defense!!.name} is neglected.")
        }
    }
}
```

`Weapon Usage in Strategy` **Code Snippet (partial code from Strategy.kt):**

```
if (general.weapon is Yin_Yang_Swords) {                                // Yin-Yang
Swords
    (general.weapon as Yin_Yang_Swords).execute(general, tar)
} else if (general.weapon is Blue_Steel_Blade) {                        // Blue Steel
Blade
    (general.weapon as Blue_Steel_Blade).execute(general, tar)
}
val originalTarHP = tar.currentHP
tar.beingAttacked(general, attackCard.suit!!)
if (tar.defense != null) {
    tar.defense!!.valid =
        true                                                             // Blue
Steel Blade: reset the defense card
}
```

Sample Output:

```
Huang Gai draw 2 card(s), now has 5 card(s). Hand: ♡ Q Peach, ♦ Q Peach, ♡ 7
Peach, ♡ J Attack, ♠ 7 Attack
Huang Gai can attack target general Liu bei.
[Blue Steel Blade] Liu bei's Defense Card: Eight Trigrams Formation is neglected.
Huang Gai spends a card ♡ J Attack to attack Liu bei.
Liu bei being attacked.
Defense card is banned.
[Eight Trigrams Formation] cannot help Liu bei dodge.
Liu bei dodge 1 attack by spending a DODGE card. Current cards: 4.
```

4. DefenseCard Factory

This is a Singleton DefenseCard Factory, called in cardGenerator from General Manager to create weapon cards.

In this factory, there is only one method called createCard overriding the super createCard method to return DefenseCards. In this method, a var defenseCardList holds all the possible weapon cards with their suit and rank (e.g., Zhuge Crossbow,Spade,A). A piece of information random is randomly selected and split by from the weaponCardList. A val card is randomly selected from the list.

A val suits is a String list holds four suits extracted from `random[1]`. A val ranks is a String list holds thirteen ranks extracted from `random[2]`.

In each call of DefenseCardFactory.createCard(), a defense card will be created and assigned with proper suit and rank.

Finally, it returns a card to cardGenerator()

DefenseCard Factory Code Snippet:

```
class DefenseFactory : CardFactory() {
    var defenseCardList: MutableList<String> = mutableListOf(
        "Eight Trigrams Formation,Spade,2", "Eight Trigrams Formation,Club,2",
    )

    override fun createCard(): Card {
        var random = defenseCardList.random()
        var information = random.split(',')
        val type = information[0]
        val suit = information[1]
        val rank = information[2]
        var card = when (type) {
            "Eight Trigrams Formation" -> Eight_Trigrams_Formation(null, null)
            else -> throw IllegalArgumentException("Invalid Weapon Card")
        }
        card.suit = suit
        card.rank = rank
        defenseCardList.remove(random)
        return card
    }
}
```

5. DefenseCard

Explanation:

In DefenseCard.kt, there are only one kind of defense card, which is `Eight Trigrams Formation`. It is inherited from DefenseCard class.

In the class, the name is overridden into corresponded card name.

Besides, a `valid` boolean variable is assigned to indicate if this defense card can take effect.

DefenseCard Code Snippet:

```
abstract class DefenseCard(initializer: General?, receiver: General?):
Card(initializer, receiver){
    open var valid: Boolean = true

    abstract fun execute(initializer: General, receiver: General?): Boolean
}
```

Sample Output:

```
♣ 2 Eight Trigrams Formation
```

The corresponding functionalities of `Eight Trigrams Formation` are shown below.

---

1. Eight Trigrams Formation:

It is a passive defense (i.e., when used, only its existence is checked to judge whether its owner has the weapon skill).

When a general needs a `Dodge` card, this defense could do a judgement before the general really spends a Dodge. If the judgement card's color is `Red` (i.e., `Heart` or `Diamond`), this defense spends a `Dodge` on behalf of this general. Otherwise, this general has to spend `Dodge` on his/her own.

For convenience, I assume the success chance of this defense skill is `0.5` because

- The cards with `Red` (i.e., `Heart` or `Diamond`) and `Black` (i.e., the rest of card) are evenly distributed.
- The deck is shuffled, forcing the `Red probability` close to 0.5.

Each time a general needs `a Dodge` or `double Dodge`, I integrate the Eight Trigrams Formation in to the real game phase.

The gaming logics with Eight Trigrams Formation are as follows:

- If the general needs two dodge, `Eight Trigrams Formation could do two judgement for him/her.` For `two-Dodge` case, if the two judgements fail, the general has to spend two Dodge cards on his/her own. Otherwise, the general only needs to spend 0 or 1 Dodge cards.
- The Eight Trigrams Formation has `highest priority` when judging whether the general has Dodge. If the judgement succeed, the general does not need to spend Dodge card.

Defense Code Snippet:

```
class Eight_Trigrams_Formation(initializer: General?, receiver: General?) :
DefenseCard(initializer, receiver){
    override val name = "Eight Trigrams Formation"
    override fun execute(initializer: General, receiver: General?): Boolean {
        if(valid == false){
            println("Defense card is banned.")
            return false
        }
        val random = (0..1).random()
        if(random <= 0.5){
            println("[Eight Trigrams Formation] of ${initializer.name}: Success!")
            return true
        }
        else{
            println("[Eight Trigrams Formation] of ${initializer.name}: Failure.")
            return false
        }
    }
}
```

`hasDodgeCard()` **Code Snippet (partial code from General.kt):**

```
open fun hasDodgeCard(): Boolean {
    if(this.defense is Eight_Trigrams_Formation){
        eight1 = (defense as Eight_Trigrams_Formation).execute(this,null)
    }
    if(eight1 == true){
        return true
    }
    else{
        for (card in hand) {
            if (card is Dodge) {
                return true
            }
        }
    }
    return false
}
```

`hasTwoDodgeCard()` **Code Snippet (partial code from General.kt):**

```
open fun hasTwoDodgeCard(): Boolean {
    var eightCount = 0
    if(this.defense is Eight_Trigrams_Formation){
        eight1 = (defense as Eight_Trigrams_Formation).execute(this,null)
        eight2 = (defense as Eight_Trigrams_Formation).execute(this,null)
    }
    if(eight1 == true){
        eightCount++
    }
    if(eight2 == true){
        eightCount++
    }

    var cnt = 0
    for (card in hand) {
        if (card is Dodge) {
            cnt++
        }
    }
    return cnt+eightCount >= 2
}
```

`beingAttacked(sender: General, suit: String)` **Code Snippet (partial code from General.kt):**

```
open fun beingAttacked(sender: General, suit: String) { //TODO: Subclass waiting
for override
```

```
    println("$name being attacked.")
    if (sender is XuChu) {
        if (sender.atk) {
            val hDC = hasDodgeCard()
            if (hDC) {
                if (eight1 == true) {
                    println("[Eight Trigrams Formation] helps ${name} dodge.")
                    eight1 = false
                } else {
                    if(defense is Eight_Trigrams_Formation){
                        println("[Eight Trigrams Formation] cannot help ${name}
dodge.")
                    }
                    dodge()
                }
            } else {
                currentHP -= 2
                if(currentHP <= 0){
                    println("$name is attacked successfully, current HP is 0")
                }else{
                    println("$name is attacked successfully, current HP is
$currentHP")
                }

            }
            if (player is Lord) {
                lord?.notifyObservers(hDC)
            }
        }
    }
    if (sender.lvbu) {
        val hDC1 = hasTwoDodgeCard()
        if (hDC1) {
            var count = 2
            if(eight1 == true){
                println("[Eight Trigrams Formation] helps ${name} dodge.")
                eight1 = false
                count--
            }
            if(eight2 == true){
                println("[Eight Trigrams Formation] helps ${name} dodge.")
                eight2 = false
                count--
            }
            if(count == 2){
                println("[Eight Trigrams Formation] cannot help ${name} dodge at
all.")
            }

            for(i in 0 until count ){
                dodge()
            }
            println("[Unrivaled] totally need to spending two doge card")
        } else {
```

```
            currentHP -= 1
            if(currentHP <= 0){
                println("$name is attacked successfully, current HP is 0")
            }else{
                println("$name is attacked successfully, current HP is
$currentHP")
            }
        }
        if (player is Lord) {
            lord?.notifyObservers(hDC1)
        }
    }else{
        val hDC = hasDodgeCard()
        if (hDC) {
            if(eight1 == true){
                println("[Eight Trigrams Formation] helps ${name} dodge.")
                eight1 = false
            }
            else{
                println("[Eight Trigrams Formation] cannot help ${name} dodge.")
                dodge()
                println("$name dodged attack by spending a DODGE card.")
}
        } else {
            currentHP -= 1
            if(currentHP <= 0){
                println("$name is attacked successfully, current HP is 0")
            }else{
                println("$name is attacked successfully, current HP is
$currentHP")
            }

        }
        if (player is Lord) {
            lord?.notifyObservers(hDC)
        }
    }
    if (currentHP <= 0) {
        alive = false
        killer = sender
        currentHP = 0
        println("$name's hp is $currentHP now!")
    }
}
```

Sample Output:

```
// Judgement Failure

Liu bei can attack target general Zhao Yun.
Liu bei spends a card ♦ 7 Attack to attack Zhao Yun.
```

```
Zhao Yun being attacked.
[Eight Trigrams Formation] of Zhao Yun: Failure.
[Eight Trigrams Formation] cannot help Zhao Yun dodge.
Zhao Yun dodge 1 attack by spending a Dodge card. Current cards: 3.


// Judgement Success

Cao Cao can attack target general Hua Xiong.
Cao Cao spends a card ♠ 9 Attack to attack Hua Xiong.
Hua Xiong being attacked.
[Eight Trigrams Formation] Success!
[Eight Trigrams Formation] helps Hua Xiong dodge.
Cao Cao (HP: 5) discarded 1 card(s), now has 5 card(s). Hand: ♡ 2 Dodge, ♡ 3
Peach, ♡ 10 Attack, ♠ 9 Attack, ♦ 4 Dodge
```

6. ShuGeneral

- Shu General class

Explanation:

This is an abstract class created for all Shu generals. The only usage of this class is `to distinguish Shu generals from other generals`.

This Shu identity could be used in `Liu Bei's Rouse Skill`. In the `GeneralFacotry.kt createRandomGeneral()`, if a Shu general is created, it will be added to the Shu Chain owned by Lord Liu Bei.

Code Snippet:

```
open class ShuGeneral(player: Player) : General(player) {
    override var name: String = ""
}
```

In the following subsection, each Shu general's Skills are explained.

- Liu Bei

Explanation:

Liu Bei is a class inherited from ShuGeneral.

Liu Bei has two Skills, one is `[Benevolence]`, which can only be used when Liu Bei is a lord. The other one is `[Rouse]`, which can be used freely.

- For `RouseJudgement()`: It judges whether the generals in Shu Chain could spend an Attack for Liu Bei. It returns a boolean value.

- For `Rouse()`: It calls the generals in Shu Chain to spend an Attack for Liu Bei. This function includes discarding card from hand and adding card to the deck. For general `Zhao Yun`, there is a special case to handle. When faced with `Barbarian` and `Duel`, this function is called.
- For `RouseAttack()`: It returns a card that a Shu general spend for Liu Bei. This function is used in `LiuBeiStrategy`. When a Shu general spends an Attack for Liu Bei, this card can be used to print attack card information.

Liu Bei class Code Snippet:

```
class LiuBei(player: Player) : ShuGeneral(player) {
    override var maxHP = 5
    override var name: String = "Liu bei"
    var shuList = mutableListOf<General>()

    fun addShu(general: General) {
        shuList.add(general)
    }

    fun removeShu(general: General) {
        shuList.remove(general)
    }

    fun checkAlive() {
        val shulistcopy = shuList.toMutableList()
        for (gen in shulistcopy) {
            if (gen.alive == false) {
                removeShu(gen)
            }
        }
    }

    fun RouseJudgement(): Boolean{
        if (shuList.size > 0) {
            println("[Rouse] Liu Bei can ask any Shu general that is in play to
use an ATTACK card for him.")
            for (gen in shuList) {
                if (gen.hasAttackCard() && gen.shouldHelpLord()) {
                    if (gen is ZhaoYun) {
                        val hand1 = gen.hand
                        for (card in hand1) {
                            if (card is Attack) {
                                println("Shu general ${gen.name} can spend an
ATTACK for Liu Bei.")
                                return true
                            }
                        }
                        for (card in hand1) {
                            if (card is Dodge) {
                                println("Shu general ${gen.name} can spend an
ATTACK for Liu Bei.")
                                return true
```

```
                            }
                        }
                    } else {
                        val hand1 = gen.hand
                        for (card in hand1) {
                            if (card is Attack) {
                                println("Shu general ${gen.name} can spend an
ATTACK for Liu Bei.")
                                return true
                            }
                        }
                    }
                }
                else{
                    println("Shu general: ${gen.name} cannot spend ATTACK card for
Liu Bei.")
                }
            }
            return false
        }
        else{
            println("There are no Shu generals in this game.")
        }
        return false
    }
    fun Rouse(): Boolean {
        if (shuList.size > 0) {
            println("[Rouse] Liu Bei can ask any Shu general that is in play to
use an ATTACK card for him.")
            for (gen in shuList) {
                if (gen.hasAttackCard() && gen.shouldHelpLord()) {
                    if (gen is ZhaoYun) {
                        val hand1 = gen.hand
                        for (card in hand1) {
                            if (card is Attack) {
                                gen.hand.remove(card)
                                discardDeck.add(card)
                                println("Shu general ${gen.name} spends an ATTACK
for Liu Bei.")
                                return true
                            }
                        }
                        for (card in hand1) {
                            if (card is Dodge) {
                                gen.hand.remove(card)
                                discardDeck.add(card)
                                println("Shu general ${gen.name} spends an ATTACK
for Liu Bei.")
                                return true
                            }
                        }
                    } else {
                        val hand1 = gen.hand
                        for (card in hand1) {
```

```
                            if (card is Attack) {
                                gen.hand.remove(card)
                                discardDeck.add(card)
                                println("Shu general ${gen.name} spends an ATTACK
for Liu Bei.")
                                return true
                            }
                        }
                    }
                }
                else{
                    println("Shu general: ${gen.name} cannot spend ATTACK card for
Liu Bei.")
                }
            }
            return false
        }
        else{
            println("There are no Shu generals in this game.")
        }
        return false
    }

    fun RouseAttack(): Card? {
        if (shuList.size > 0) {
            println("[Rouse] Liu Bei can ask any Shu general that is in play to
use an ATTACK card for him.")
            for (gen in shuList) {
                if (gen.hasAttackCard() && gen.shouldHelpLord()) {
                    if (gen is ZhaoYun) {
                        val hand1 = gen.hand
                        for (card in hand1) {
                            if (card is Attack) {
                                gen.hand.remove(card)
                                discardDeck.add(card)
                                println("Shu general ${gen.name} spends an ATTACK
for Liu Bei.")
                                return card
                            }
                        }
                        for (card in hand1) {
                            if (card is Dodge) {
                                gen.hand.remove(card)
                                discardDeck.add(card)
                                println("Shu general ${gen.name} spends an ATTACK
for Liu Bei.")
                                return card
                            }
                        }
                    } else {
                        val hand1 = gen.hand
                        for (card in hand1) {
                            if (card is Attack) {
                                gen.hand.remove(card)
```

```
                                discardDeck.add(card)
                                println("Shu general ${gen.name} spends an ATTACK
for Liu Bei.")
                                return card
                            }
                        }
                    }
                }
                else{
                    println("Shu general: ${gen.name} cannot spend ATTACK card for
Liu Bei.")
                }
            }
        }
        else{
            println("There are no Shu generals in this game.")
        }
        return null
    }

    override fun hasAttackCard(): Boolean {
        checkAlive()

        if(shuList.size > 0){
            for (gen in shuList) {
                if (gen.hasAttackCard() && gen.shouldHelpLord())
                    return true
            }
        }
        for (card in hand) {
            if (card is Attack) {
                return true
            }
        }
        return false
    }

    override fun spendAttackCard() {
        if (lvbu) {
            for (i in 0..1) {
                val rouse = Rouse()
                if (rouse == false) {
                    var isSpent = false
                    var hand1 = hand.toMutableList()
                    for (card in hand1) {
                        if (!isSpent && card is Attack) {
                            hand.remove(card)
                            discardDeck.add(card)
                            isSpent = true
                        }
                    }
                }
            }
        } else {
```

```
            val rouse = Rouse()
            if (rouse == false) {
                var isSpent = false
                var hand1 = hand.toMutableList()
                for (card in hand) {
                    if (!isSpent && card is Attack) {
                        hand1.remove(card)
                        discardDeck.add(card)
                        isSpent = true
                    }
                }
                hand = hand1
            }
        }
    }
}
```

- `[Benevolence]` is the first skill of Liu Bei. There are two states of Liu Bei, namely `HealthyState`and `UnhealthyState`. At the beginning of his turn, he will check if his HP is less than 2. If so, Liu Bei will be in `UnhealthyState` and then recovery 1 HP by discarding two random card. It is important to keep a safe HP in a game, so we define Liu Bei to automatically use `[Benevolence]` no matter what cards he has. However, if Liu Bei has less than 2 cards, `[Benevolence]` fails to take effect due to insufficient cards. If Liu Bei has more than 1 HP (>1), he will not use `[Benevolence]` skill.

  Code Snippet:

```
interface State {
    fun playNextCard(){}
}

class HealthyState(val strategy:LiuBeiStrategy): State{
    override fun playNextCard() {
        if(strategy.general.currentHP<2){
            strategy.state = UnhealthyState(strategy)
            strategy.state.playNextCard()
        }
        else{
            println("${strategy.general.name} is healthy.")
        }
    }
}

class UnhealthyState(val strategy:LiuBeiStrategy): State{
    override fun playNextCard() {
        if(strategy.general.currentHP>=2){
            strategy.state = HealthyState(strategy)
            println("${strategy.general.name} is now healthy.")
        }
        else {
            if(strategy.general.hand.size >= 2){
```

```
                println("${strategy.general.name} is not healthy")
                val beforeHand = strategy.general.hand.size
                val beforeHP = strategy.general.currentHP
                strategy.general.discard(2)
                strategy.general.currentHP++
                val afterHand = strategy.general.hand.size
                val afterHP = strategy.general.currentHP
                println("[Benevolence] ${strategy.general.name} gives away 2
cards (from ${beforeHand} to ${afterHand}) and recover 1 HP (from
${beforeHP} to ${afterHP}.")
            }
            else{
                println("[Benevolence] ${strategy.general.name} fails to
recover 1 HP due to insufficient cards.")
            }
            if(strategy.general.currentHP>=2){
                strategy.state = HealthyState(strategy)
                println("${strategy.general.name} leaves unhealthy state.")
                strategy.state.playNextCard()
            }
        }
    }
}
```

Sample Output:

```
// Healthy State

Liu bei draw 2 card(s), now has 6 card(s). Hand: ♡ J Attack, ♦ 2 Dodge, ♣ 7
Attack, ♡ 10 Attack, ♡ 10 Attack, ♡ 5 Kirin Bow
Liu bei is healthy.
Liu bei takes the weapon: [Kirin Bow] from his/her hand to the weapon area.
Liu bei can attack target general Zhao Yun.
....


// Unhealthy State

Liu bei draw 2 card(s), now has 2 card(s). Hand: ♣ Q Dismantle, ♣ 6 Acedia
Liu bei is not healthy
Liu bei discarded 2 card(s), now has 0 card(s). Hand:
[Benevolence] Liu bei gives away 2 cards (from 2 to 0) and recover 1 HP
(from 1 to 2).
Liu bei leaves unhealthy state.
Liu bei is healthy.
```

- `[Rouse]` is the second skill of Liu Bei. Every Shu general can spend Attack for Lord Liu Bei.

  `Remarks: Only when Liu Bei is lord, he can use this Skill.`

Liu Bei maintains a `shuList` mutable list, which is different from the `linked list` in WeiChain. When a Shu general is created, he/she will be added to the lord Liu Bei's `shuList`.

However, his Shu generals could not choose to help because of their identity (`shouldHelpLord()`) or lack of Attack card.

When

- Liu Bei is faced with `Barbarian`, he will call his Shu general to spend an Attack for him.

  Sample Output:

  ```
  Zhang Fei draw 2 card(s), now has 5 card(s). Hand: ♡ 8 Peach, ♠ 6
  Barbarians, ♡ A RainArrows, ♣ Q Dismantle, ♣ 6 Acedia
  Zhang Fei launch a command Barbarian, every one need to spend an attack
  card to avoid the attack
  [Rouse] Liu Bei can ask any Shu general that is in play to use an
  ATTACK card for him.
  Shu general: Zhang Fei cannot spend ATTACK card for Liu Bei.
  Liu bei successfully dodge the Barbarian
  Diao Chan is hurt by Barbarians, current HP is 1.
  Zhao Yun successfully dodge the Barbarian
  ```

- Liu Bei is in `Duel`, he will call his Shu general to spend an Attack for him.

  Sample Output:

  ```
  Liu bei launch a duel with Zhang Fei
  Zhang Fei launch a duel with Liu bei
  [Rouse] Liu Bei can ask any Shu general that is in play to use an
  ATTAcK card for him.
  Shu general: Zhang Fei cannot spend ATTACK card for Liu Bei.
  iu bei launch a duel with Zhang Fei
  Zhang Fei launch a duel with Liu bei
  [Rouse] Liu Bei can ask any Shu general that is in play to use an
  ATTAcK card for him.
  Shu general: Zhang Fei cannot spend ATTAcK card for Liu Bei.
  iu bei launch a duel with Zhang Fei
  Zhang Fei launch a duel with Liu bei
  [Rouse] Liu Bei can ask any Shu general that is in play to use an
  ATTAcK card for him.
  Shu general: Zhang Fei cannot spend ATTACK card for Liu Bei.
  iu bei launch a duel with Zhang Fei
  Zhang Fei lose the duel, current Hp is 3
  Liu bei doesn't have an attack card.
  ```

- Liu Bei is going to initialize an Attack to other generals in his turn.

Sample Output:

```
// Zhang Fei is a Rebel

Liu Bei can attack general Hua tuo.
[Yin-Yang Swords] fails to take effect.
[Rouse] Liu Bei can ask any Shu general that is in play to use an ATTACK
card for him.
Shu general: Zhang Fei cannot spend ATTACK card for Liu Bei.
Liu bei spends a card ♦ 8 Attack to attack Hua tuo.
Hua tuo being attacked.
Hua tuo is attacked successfully, current HP is 0
```

- Zhang Fei

Explanation:

Zhang Fei is a class inherited from ShuGeneral.

Zhang Fei only has one skill, which is `[Berserk]`. It enables Zhang Fei to spend as many `Attack` cards as he wants. In our game, it is a precious skill same as `Zhuge Crossbow`, which can make him win faster. So, when it is Zhang Fei's turn to attack, we let him spend all the Attack cards.

Zhang Fei's skill in integrated into the strategy. Therefore, its initialization is simple.

Zhang Fei class Code Snippet:

```
class ZhangFei(player: Player) : ShuGeneral(player) {
    override var maxHP = 4
    override var name: String = "Zhang Fei"
}
```

- [Berserk] is the only Skill that Zhang Fei has. This skill is judged and used in `LoyalistStrategy`and `RebelStrategy`.

  Code Snippet:

```
// Partially extracted from LoyalistStrategy. The code also applies in
RebelStrategy.

// ****************************************** SET ATTACK TIMES
  var attackingNum = 0                                                //
default
  if (general.weapon is Zhuge_Crossbow) {                             //
Zhuge Crossbow
      (general.weapon as Zhuge_Crossbow).execute(general, null)
      attackingNum =
```

```
            general.hand.size                                   // Zhuge
Crossbow: maximum number of attacks = cards in hand
  } else {
      attackingNum = 1
  }
  if (general is ZhangFei) {
      println("[Berserk] Zhang Fei can use as many ATTACK cards as he wishes
during the turn.")
      attackingNum =
          general.hand.size                                   // Zhang Fei's
Skill: maximum number of attacks = cards in hand
  }
// ****************************************** SET ATTACK TIMES
```

Sample Output:

```
[Berserk] Zhang Fei can use as many ATTACK cards as he wishes during the
turn.
Zhang Fei can attack target general Zhao Yun.
Zhang Fei spends a card ♠ 9 Attack to attack Zhao Yun.
Zhao Yun being attacked.
Zhao Yun dodge 1 attack by spending a Dodge card. Current cards: 3.
Zhao Yun dodged attack by spending a DODGE card.
Zhang Fei can attack target general Zhao Yun.
Zhang Fei spends a card ♣ 3 Attack to attack Zhao Yun.
Zhao Yun being attacked.
Zhao Yun dodge 1 attack by spending a Dodge card. Current cards: 2.
Zhao Yun dodged attack by spending a DODGE card.
```

- Zhao Yun

Explanation:

Zhao Yun is a class inherited from ShuGeneral.

Zhao Yun only has one skill, which is `[Dragon Heart]`. For him, `Attack` and `Dodge` can be used interchangeably.

Specifically, the `hasDodgeCard()`, `hasAttackCard()`, `hasTwoDodgeCard()`, `spendAttackCard()`, `Dodge()` and `spendDodgeCard()` are modified to integrate this general skill.

For `hasDodgeCard()`, `hasAttackCard()` and `hasTwoDodgeCard()`, Zhao Yun first check if he has required card. If he has the required card(s), he simply spends the cards. However, if he `does not have`/ `does not have sufficient` cards, he will check whether he has `Attack to replace Dodge`/ `Dodge to replace Attack`. In these situation, his `[Dragon Heart]`skill is activated.

For `spendAttackCard()`, `Dodge()` and `spendDodgeCard()`, the logic is same as the above. First spend the required card(s), and then spend the counterpart card(s) if he has.

In our game, we let Zhao Yun to always use this skill if necessary.

Zhao Yun class Code Snippet:

```
class ZhaoYun(player: Player) : ShuGeneral(player) {
    init {
        initializeIdentityStrategy()
    }
    override var maxHP = 4
    override var name: String = "Zhao Yun"

    fun initializeIdentityStrategy() {// For generals who have special strategy,
this helps
        // implement their action that is affected by their
        //identity instead of their general skill
        strategy = ZhaoYunStrategy(this)
        var stra: Strategy? = null
        if (player is Rebel) {
            stra = RebelStrategy(this)
        } else if (player is Lord) {
            stra = LoyalistStrategy(this)
        } else if (player is Loyalist) {
            stra = LoyalistStrategy(this)
        } else if (player is Spy) {
            if ((player as? Spy)!!.isRevealed == false) {
                stra = LoyalistStrategy(this)
            } else {
                stra = RebelStrategy(this)
            }
        }


        if (stra != null) {
            (strategy as ZhaoYunStrategy).identityStrategy = stra
        }
    }

    override fun hasDodgeCard(): Boolean {
        if (defense is Eight_Trigrams_Formation) {
            eight1 = (defense as Eight_Trigrams_Formation).execute(this, null)
        }
        if (eight1 == true) {
            return true
        }

        var activateSkill = true
        var result = false
        for (card in hand) {
            if (card is Dodge) {
                activateSkill = false
                result = true
            }
```

```
        }
        if (activateSkill == true) {
            for (card in hand) {
                if (card is Attack) {
                    result = true
                }
            }
        }
        return result
    }

    override fun hasAttackCard(): Boolean {
        var activateSkill = true
        var result = false
        for (card in hand) {
            if (card is Attack) {
                activateSkill = false
                result = true
            }
        }
        if (activateSkill == true) {
            for (card in hand) {
                if (card is Dodge) {
                    result = true
                }
            }
        }
        return result
    }

    override fun hasTwoDodgeCard(): Boolean {
        var eightCount = 0
        if (defense is Eight_Trigrams_Formation) {
            eight1 = (defense as Eight_Trigrams_Formation).execute(this, null)
            eight2 = (defense as Eight_Trigrams_Formation).execute(this, null)
        }
        if (eight1 == true) {
            eightCount++
        }
        if (eight2 == true) {
            eightCount++
        }

        var cnt = 0
        for (card in hand) {
            if (card is Dodge) {
                cnt++
            }
        }
        if (cnt < 2) {
            for (card in hand) {
                if (card is Attack) {
                    cnt++
                }
```

```
            }
        }
        return cnt + eightCount >= 2
    }

    override fun spendAttackCard() {
        if (lvbu) {
            println("[unrivaled] totally need to spending two attack card")
            val hand1 = hand.toMutableList()
            var attackCount = 0
            for (card in hand1) {
                if (card is Attack) {
                    hand.remove(card)
                    discardDeck.add(card)
                    attackCount++
                    if (attackCount == 2) {
                        break
                    }
                }
            }
            if (attackCount < 2) {
                println("[Dragon Heart] Zhao Yun's ATTACK and DODGE cards can be
used interchangeably.")
                for (card in hand1) {
                    if (card is Dodge) {
                        hand.remove(card)
                        discardDeck.add(card)
                        attackCount++
                        if (attackCount == 2) {
                            break
                        }
                    }
                }
            }
        } else {
            val hand1 = hand.toMutableList()
            var activateSkill = true
            for (card in hand1) {
                if (card is Attack) {
                    hand.remove(card)
                    discardDeck.add(card)
                    activateSkill = false
                    break
                }
            }
            if (activateSkill == true) {
                for (card in hand1) {
                    if (card is Dodge) {
                        println("[Dragon Heart] Zhao Yun's ATTACK and DODGE cards
can be used interchangeably.")
                        hand.remove(card)
                        discardDeck.add(card)
                        break
                    }
```

```
                }
            }
        }
    }

    override fun dodge() {
        if (eight1 == true) {
            println("[Eight Trigrams Formation] helps ${name} dodge.")
            return
        }

        var hasDodge = false

        var dodgeCard: Card = Dodge(null, null)
        for (card in hand) {
            if (card is Dodge) {
                dodgeCard = card
                hasDodge = true
                break
            }
        }

        if (hasDodge == true) {
            hand.remove(dodgeCard)
            discardDeck.add(dodgeCard)
            println("$name dodge 1 attack by spending a Dodge card. Current cards:
${hand.size}.")
        } else {
            var attackCard: Card = Attack(null, null)
            for (card in hand) {
                if (card is Attack) {
                    println("[Dragon Heart] Zhao Yun's ATTACK and DODGE cards can
be used interchangeably.")
                    attackCard = card
                    break
                }
            }
            hand.remove(attackCard)
            discardDeck.add(attackCard)
            println("$name dodge 1 attack by spending an ATTACK card. Current
cards: ${hand.size}.")
        }
    }

    override fun spendDodgeCard() {
        var dodgeCard: Card = Dodge(null, null)
        var activateSkill = true
        for (card in hand) {
            if (card is Dodge) {
                dodgeCard = card
                activateSkill = false
            }
        }
        hand.remove(dodgeCard) //not yet prove
```

```
            discardDeck.add(dodgeCard)

            var attackCard: Card = Attack(null, null)
            if (activateSkill == true) {
                for (card in hand) {
                    if (card is Attack) {
                        println("[Dragon Heart] Zhao Yun's ATTACK and DODGE cards can
be used interchangeably.")
                        attackCard = card
                    }
                }
                hand.remove(attackCard)
                discardDeck.add(attackCard)
            }
            println("$name spend a DODGE card ${suitMap[dodgeCard.suit]}
${dodgeCard.rank} ${dodgeCard.name} to avoid a defect.")
        }
    }
```

`[Dragon Heart]` is the only skill of Zhao Yun. For him, `Attack` and `Dodge` can be used interchangeably. The codes are integrated into the `ZhaoYunStrategy` above. However, there are several scenarios to show Zhao Yun's skill.

- `Attack` used as `Dodge`.

  Sample Output:

```
---Distribute Cards for the Starting Hand---
Cao Cao draw 4 card(s), now has 4 card(s). Hand: ♠ 7 Attack, ♣ K Negate, ♣ 5
Green Dragon Blade, ♠ A Zhuge Crossbow
Zhao Yun draw 4 card(s), now has 4 card(s). Hand: ♣ A Duel, ♠ 3 Dismantle, ♦
7 Attack, ♡ 2 Dodge

            // Useless part
            // Sun Shang Xiang draw 4 card(s), now has 4 card(s). Hand: ♡ 2
Dodge, ♦ A Duel, ♡ 2 Dodge, ♡ A RainArrows
            // Hua tuo draw 4 card(s), now has 4 card(s). Hand: ♡ 2 Dodge,
♡ K Dodge, ♣ 2 Yin-Yang Swords, ♠ 2 Eight Trigrams Formation
            // Lv Bu draw 4 card(s), now has 4 card(s). Hand: ♣ 6 Attack, ♣
3 Attack, ♡ 5 Kirin Bow, ♦ 5 Dodge
            // Zhang Fei draw 4 card(s), now has 4 card(s). Hand: ♡ J
Attack, ♡ 3 Peach, ♣ 4 Dismantle, ♦ 9 Dodge

---Turn 1---
Cao Cao draw 2 card(s), now has 6 card(s). Hand: ♠ 7 Attack, ♣ K Negate, ♣ 5
Green Dragon Blade, ♠ A Zhuge Crossbow, ♠ 9 Attack, ♠ 6 Acedia
Cao Cao takes the weapon: [Green Dragon Blade] from his/her hand to the
weapon area.

            // Useless part
            // Cao Cao set a judgement Acedia to Zhang Fei
```

```
Cao Cao can attack target general Zhao Yun.
Cao Cao spends a card ♠ 7 Attack to attack Zhao Yun.
Zhao Yun being attacked.
Zhao Yun dodge 1 attack by spending a Dodge card. Current cards: 3.
[Green Dragon Blade] "Attack" enabled again.
Cao Cao can attack target general Zhao Yun.
Cao Cao spends a card ♠ 9 Attack to attack Zhao Yun.
Zhao Yun being attacked.
[Dragon Heart] Zhao Yun's ATTACK and DODGE cards can be used
interchangeably. (Attack for Dodge)
Zhao Yun dodge 1 attack by spending an ATTACK card. Current cards: 2.
```

- `Dodge` used as `Attack`.

  Sample Output:

```
Zhao Yun draw 2 card(s), now has 4 card(s). Hand: ♣ Q Dismantle, ♦ Q Peach,
♡ 2 Dodge, ♡ 2 Dodge
Zhao Yun dismantle a card ♦ 9 Attack from Hua Xiong
(Distance) Zhao Yun cannot attack target general Huang Gai.

            // Useless part
            // Lv Bu judging the Acedia card.
            // Lv Bu is unlucky,need to skip a turn
            // Lv Bu draw 2 card(s), now has 5 card(s). Hand: ♦ K Violet
Stallion, ♣ 2 Yin-Yang Swords, ♡ 5 Kirin Bow, ♦ 10 Dodge, ♠ 4 StealingSheep
            // Lv Bu takes the horse: [Violet Stallion] from his/her hand to
the horse area.
            // Lv Bu takes the weapon: [Yin-Yang Swords] from his/her hand
to the weapon area.
            // Lv Bu steal a card Attack from Hua tuo. Hand: ♡ 5 Kirin Bow,
♦ 10 Dodge, ♠ 4 StealingSheep, ♡ J Attack
            // Lv Bu doesn't have an attack card.

Huang Gai draw 2 card(s), now has 4 card(s). Hand: ♠ 8 Attack, ♦ A Duel, ♦ 7
Dodge, ♡ 7 Peach
Huang Gai launch a duel with Zhao Yun
[Dragon Heart] Zhao Yun's ATTACK and DODGE cards can be used
interchangeably. (Dodge for Attack)
Zhao Yun launch a duel with Huang Gai
Huang Gai launch a duel with Zhao Yun
[Dragon Heart] Zhao Yun's ATTACK and DODGE cards can be used
interchangeably. (Dodge for Attack)
Zhao Yun launch a duel with Huang Gai
Huang Gai lose the duel, current Hp is 2
```