

Refactoring after build #2

<i>List of potential refactoring targets</i>		<i>Reason for choosing</i>
1	In class “CardExchangeView”, use the java build-in “observer function”, set the class to be observer, to observe class “Player” (method “removeCard()”, “getNewCard()”, to replace the observer function we created	To increase compatibility
2	In class “AttackController”, consolidate Duplicate Conditional Fragments	To improve code reusability and understandability.
3	In class “Card” of game model, extract the “cardType” to be a separate class	To improve reusability and lower coupling
4	Separate SetupView from Reinforcement View to decrease coupling	
5	Move inner class “PhaseState” out of class “GameState” to decrease coupling	
6	In Attack phase, create a new method in the player class for the attack action	

Refactoring operations applied:

1. In class “CardExchangeView”, use the java build-in “observer function”, set the class to be observer, to observe class “Player” (method “removeCard()”, “getNewCard()”, to replace the observer function we created, to increase compatibility.

The screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer lists several files, with 'src\views\game\BaseObserverFrame.java' selected. The code editor displays the following Java code:

```

@@ -1,22 +0,0 @@
1  -package views.game;
2  -
3  -import javax.swing.JFrame;
4  -
5  -import models.game.Player;
6  -
7  -
8  -/**
9  - * The Class BaseObserverFrame.
10 - * @author Bingyang
11 - * @version 2.0
12 - */
13 -public abstract class BaseObserverFrame extends JFrame {
14 -
15 -    /** The player. */
16 -    protected Player player;
17 -
18 -    /**
19 -     * Update.
20 -     */
21 -    public abstract void update();
22 -}

```

src\models\game\Player.java	29	32	*/
src\views\game\BaseObserverFrame.java	30		-public class CardExchangeView extends BaseObserverFrame {
src\views\game\CardExchangeView.java	31	33	+public class CardExchangeView extends JFrame implements Observer {
src\views\game\ReinforcementView.java	32	34	/** The main panel. */
	33	35	private JPanel mainPanel = new JPanel();
		36	@@ -44,27 +47,27 @@ public class CardExchangeView extends BaseObserverFrame {
	44	47	/** The toggle buttons. */
	45	48	List<JToggleButton> toggleButtons;
	46	49	
	47		- /** The exchange button. */
		50	+ /** The exchange button. */
	48	51	private JButton exchangebutton;
	49	52	
	50		- /** The original view. */
		53	+ /** The original view. */
	51	54	ReinforcementView originalview;
	52	55	
	53		- /** The return button. */
		56	+ /** The return button. */
	54	57	private JButton returnbutton=new JButton("Return");
	55	58	
src\controller...\ReinforcementController.java	21	22	*/
src\models\game\Player.java	22		-public class Player {
src\views\game\BaseObserverFrame.java	23	24	+public class Player extends Observable {
src\views\game\CardExchangeView.java	24	25	/** The id. */
src\views\game\ReinforcementView.java	25	26	private int id;
			@@ -43,14 +44,7 @@ public class Player {
	43	44	// this.setStrategy(strategy);

Test:

```

/**
 * This test case tests calculation of number of armies in exchange with cards
 */
@Test
public void testExchangeArmyForCard() {
    int expectedResult = 15;
    Card c1=new Card(player1);
    Card c2=new Card(player1);
    Card c3=new Card(player1);
    c1.setCardType(CardType.ARTILLERY);
    c2.setCardType(CardType.CAVALRY);
    c3.setCardType(CardType.INFANTRY);

    List<Card> exchangeList=new ArrayList<Card>();
    exchangeList.add(c1);
    exchangeList.add(c2);
    exchangeList.add(c3);

    int result = player1.exchangeCardforArmy(exchangeList);
    assertEquals(expectedResult, result);

    expectedResult = 5;
    Card c6=new Card(player2);
    Card c4=new Card(player2);
    Card c5=new Card(player2);
    c6.setCardType(CardType.ARTILLERY);
    c4.setCardType(CardType.CAVALRY);
    c5.setCardType(CardType.INFANTRY);
    List<Card> exchangeList2=new ArrayList<Card>();

    exchangeList.add(c4);
    exchangeList.add(c5);
    exchangeList.add(c6);

    result = player2.exchangeCardforArmy(exchangeList2);
    assertEquals(expectedResult, result);
}

```

2. In class “AttackController”, consolidate Duplicate Conditional Fragments to improve code reusability and understandability.

```

70 | 71 | case AttackView.ContinueStr:
71 | - |     Country fromCountry = attackView.getSelectedCountryFrom();
72 | - |     Country toCountry = attackView.getSelectedCountryTo();
73 | - |     int diceNumber = attackView.getAttacherDiceNumber();
74 | - |
75 | - |     if (GameState.getInstance().getCurrentPlayer().conquer(toCountry)) {
76 | - |         if (GameState.getInstance().getMap().mapOwner(GameState.getInstance().getCurrentPlayer()))
77 | - |             StateView.getInstance().showEndGameView();
78 | - |         else
79 | - |             attackView.showMoveArmiesState(diceNumber);
80 | - |     }
81 | - |     else {
82 | - |         if (GameState.getInstance().getCurrentPlayer().getArmyNumber() == 0) {
83 | - |             // current player ended his/her turn.
84 | - |             GameState.getInstance().endPlayerTurn();
85 | - |             StateView.getInstance().getMapPanel().addCountryTableForMap(GameState.getInstance().getMap());
86 | - |
87 | - |             GameState.getInstance().setPhase(Phase.REINFORCEMENT);
88 | - |             StateView.getInstance().showReinforcementView();
89 | - |         }
90 | - |         else {
91 | - |             if (GameState.getInstance().getCurrentPlayer().isAttackPossible())
92 | - |                 attackView.showSelectionState();

```

```

123 | - |
124 | - |     //if(defenderCountry.getNumOfArmies() == 0) {
125 | - |     if (GameState.getInstance().getCurrentPlayer().conquer(defenderCountry)) {
126 | - |         //defenderCountry.setOwner(GameState.getInstance().getCurrentPlayer());
127 | - |         if (GameState.getInstance().getMap().mapOwner(GameState.getInstance().getCurrentPlayer()))
128 | - |             StateView.getInstance().showEndGameView();
129 | - |         else
130 | - |             attackView.showMoveArmiesState(diceNo);
131 | - |     }
132 | - |     else {
133 | - |         if (GameState.getInstance().getCurrentPlayer().getArmyNumber() == 0) {
134 | - |             // current player ended his/her turn.
135 | - |             GameState.getInstance().endPlayerTurn();
136 | - |             StateView.getInstance().getMapPanel().addCountryTableForMap(GameState.getInstance().getMap());
137 | - |
138 | - |             GameState.getInstance().setPhase(Phase.REINFORCEMENT);
139 | - |             StateView.getInstance().showReinforcementView();
140 | - |         }
141 | - |         else {
142 | - |             if (GameState.getInstance().getCurrentPlayer().isAttackPossible())
143 | - |                 attackView.showSelectionState();
144 | - |             else {
145 | - |                 GameState.getInstance().setPhase(Phase.FORTIFICATION);
146 | - |                 StateView.getInstance().showFortificationView();
147 | - |             }
148 | - |         }
149 | - |     }

```

```

125 + private void checkNextStep() {
126 +     if (GameState.getInstance().getCurrentPlayer().conquer(defenderCountry)) {
127 +         if (GameState.getInstance().getMap().mapOwner(GameState.getInstance().getCurrentPlayer()))
128 +             StateView.getInstance().showEndGameView();
129 +         else
130 +             attackView.showMoveArmiesState(fromCountry.getNumOfArmies()<=diceNumber ? diceNumber-1 : diceNu
mber);
131 +             attackView.showMoveArmiesState(1);
132 +     }
133 +     else {
134 +         // in case that attacker lost the country
135 +         if (defenderCountry.getNumOfArmies() == 0) {
136 +             defenderCountry.setOwner(attackerCountry.getOwner());
137 +             attackerCountry.getOwner().moveArmies(defenderCountry, attackerCountry, 1);
138 +         }
139 +
140 +         if(GameState.getInstance().getCurrentPlayer().getArmyNumber() == 0) {
141 +             // current player ended his/her turn.
142 +             GameState.getInstance().endPlayerTurn();
143 +             GameState.getInstance().setPhase(Phase.REINFORCEMENT);
144 +             StateView.getInstance().showReinforcementView();
145 +         }
146 +         else {
147 +             if(GameState.getInstance().getCurrentPlayer().isAttackPossible())
148 +                 attackView.showSelectionState();

```

```

/**
 * test for attack of Conquer
 */
@Test
public void testAttackConquer(){
    int ArmyNumberDefenderCountry = defenderCountry.getNumOfArmies();

    // Player0 attacks player1, until player1 has no army left in the defenderCountry
    defenderCountry.removeArmies(ArmyNumberDefenderCountry);
    player0.conquer(defenderCountry);
    assertTrue(defenderCountry.getOwner()==player0);
}

/**
 * test for attack of valid move after conquering
 */
@Test
public void testValidMoveAfterConquering(){
    int ArmyNumberDefenderCountry = defenderCountry.getNumOfArmies();

    // Player0 attacks player1, until player1 has no army left in the defenderCountry
    defenderCountry.removeArmies(ArmyNumberDefenderCountry);
    player0.conquer(defenderCountry);
    player0.moveArmies(attackerCountry, defenderCountry, 5);

    assertTrue(defenderCountry.getNumOfArmies()==5);
}

```

3. In class “Card” of game model, extract the “cardType” to be a separate class to improve reusability and lower coupling.

save\201811291218.save	+			@@ -15,42 +15,6 @@ public class Card {
src\controllers\ga...\GameStartController.java	●	15	15	/** The card type. */
src\controllers\game\MenuController.java	+	16	16	private CardType cardType;
src\models\game\Card.java	■	17	17	
src\models\game\CardType.java	+	18		- /**
src\models\game\GameState.java	●	19		- * The Enum CardType.
src\models\map\Continent.java	●	20		- */
src\test\models\ga...\ReinforcementTests.java	●	21		- public enum CardType {
src\views\game\CardExchangeView.java	●	22		-
src\views\game\ReinforcementView.java	●	23		- /** The infantry. */
src\views\game\StateView.java	●	24		- INFANTRY(0),
		25		-
		26		- /** The cavalry. */
		27		- CAVALRY(1),
		28		-
		29		- /** The artillery. */
		30		- ARTILLERY(2);
		31		-
		32		- /** The card type code. */
		33		- private final int cardTypeCode;
		34		-
		35		- /**
		36		- * Instantiates a new card type.
		37		- *
		38		- * @param cardTypeCode the card type code
		39		- */
		40		- private CardType(int cardTypeCode) {

save\201811291218.save	+			@@ -0,0 +1,33 @@
src\controllers\ga...\GameStartController.java	●		1	+package models.game;
src\controllers\game\MenuController.java	+		2	+
src\models\game\Card.java	●		3	+public enum CardType {
src\models\game\CardType.java	+		4	+ /** The infantry. */
src\models\game\GameState.java	●		5	+ INFANTRY(0),
src\models\map\Continent.java	●		6	+
src\test\models\ga...\ReinforcementTests.java	●		7	+ /** The cavalry. */
src\views\game\CardExchangeView.java	●		8	+ CAVALRY(1),
src\views\game\ReinforcementView.java	●		9	+
src\views\game\StateView.java	●		10	+ /** The artillery. */
			11	+ ARTILLERY(2);
			12	+
			13	+ /** The card type code. */
			14	+ private final int cardTypeCode;
			15	+
			16	+ /**
			17	+ * Instantiates a new card type.
			18	+ *
			19	+ * @param cardTypeCode the card type code
			20	+ */
			21	+ private CardType(int cardTypeCode) {
			22	+ this.cardTypeCode = cardTypeCode;
			23	+ }
			24	+
			25	+ /**
			26	+ * Gets the card type code.

Test:

```

/**
 * This class tests card settings of the game
 * @author Lynn
 *
 */
public class CardTest {
    String cardTypeStr = "INFANTRY";
    String CCardTypeStr;

    /**
     * Set up before test
     */
    @Before
    public void setUp(){
        GameState.reset();
        Player player0 = new Player();
        GameState.getInstance().getPlayerList().add(player0);
        Card c = new Card(player0);
        c.setCardType(CardType.INFANTRY);
        CCardTypeStr = c.getCardType().toString();
    }

    @Test
    public void test() {
        assertTrue(cardTypeStr == CCardTypeStr);
    }
}

```