โครงการ

เรื่อง RASRI-TAIRE

วิชา CP241 Data Structure

จัดทำโดย

นาย.ภูริปกรณ์ ศรียอด 61102010158

นาย.วรวิทย์ นาคนาวา 61102010160

นาย.กานต์ชนิต โพธิสุวรรณ 61102010419

เสนอ

อาจารย์.วีรยุทธ เจริญเรืองกิจ

และ

อาจารย์.ศุภชัย ไทยเจริญ

คณะวิทยาศาสตร์ ภาควิชาวิทยาการคอมพิวเตอร์

มหาวิทยาลัยศรีนครินทรวิโรฒ

ภาคเรียนที่2 ปี การศึกษา 2562

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา CP241 DATA STRUCTURE โดยมีจุดประสงค์ เพื่อให้รู้ถึงชนิดข้อมูลแบบต่างๆ ที่เป็นพื้นฐานที่จำเป็นสำหรับการเรียนในวิชาอื่นๆ ที่สูงขึ้นไปที่เกี่ยวข้องกับสาขาวิทยาการคอมพิวเตอร์ และเป็นชนิดข้อมูลที่ไม่ได้เป็นชนิดข้อมูลมาตรฐานที่มากับโปรแกรมภาษาสูง ชนิดข้อมูลที่ศึกษาได้แก่ สแตก คิว ลิสต์ ซึ่งมีโครงสร้างข้อมูลแบบเชิงเส้น (linear) ต้นไม้ซึ่งมีโครงสร้างแบบลำดับชั้น (hierarchical) และกราฟซึ่งมีโครงสร้างข้อมูลแบบเครือข่าย (network) เพื่อศึกษาหลักการของวิธีเรียงลำดับข้อมูล และหลักการของการค้นหาข้อมูล ซึ่งเป็นกิจกรรมที่สำคัญมากในการประยุกต์การใช้งานกับข้อมูล เพื่อนำเสนอการศึกษาชนิดข้อมูลและโครงสร้างข้อมูลแบบต่าง ๆ ด้วยหลักการของชนิดข้อมูลแบบนามธรรม (Abstract Data Type -ADT) ซึ่งเป็นรากฐานที่สำคัญของการโปรแกรมเชิงวัตถุ (Object Oriented Programming) เป็นต้น

ในการจัดทำรายงานประกอบสื่อการเรียนรู้ในครั้งนี้ทางคณะผู้จัดทำขอขอบคุณ อ.วีรยุทธ เจริญเรืองกิจ และ อาจารย์.ศุภชัย ไทยเจริญ อาจารย์ประจำวิชาผู้ให้ความรู้และแนวทางการศึกษาโดยทางคณะผู้จัดทำทุกคนหวังว่าโครงการเรื่องนี้จะสามารถนำไปใช้ต่อยอดให้เกิดประโยชน์ให้กับผู้อ่านต่อได้ในอนาคต ขอขอบคุณมา ณ ที่นี้ด้วย ขอบคุณครับ

คณะผู้จัดทำ

สารบัญ

บทนำ

อธิบายเหตุผลว่าแอ็พพลิเคชันนี้เป็นประโยชน์ต่อคุณหรือต่อผู้อื่นอย่างไร

แอ็พพลิเคชันที่ได้ทำ เป็นเกมเรียงไพ่คล้ายๆกับ solitaire โดยจะเปลี่ยนเป็นการนำเสนอในรูปแบบจักราศีแทนซึ่งจะเป็นการแบ่งหมวดไพ่ในสำรับทั้ง 52 ใบ เป็น 4 หมวดคือ โพดำ(♠) โพแดง(♥) ข้าวหลามตัด(♦) และดอกจิก(♣) โดยจะให้เรียงจาก ราศีมังกร ราศีกุมภ์ ราศีมีน ราศีเมษ ราศีพฤษภ ราศีเมถุน ราศีกรกฎ ราศีสิงห์ ราศีกันย์ ราศีตุล ราศีพิจิก ราศีธนู ราศีคนแบกงู ซึ่งเกมของเราจะเป็นเกมที่ช่วยในเรื่องการฝึกสมองด้านความจำ ไหวพริบ การวางแผนต่างๆ และยังเป็นการทดสอบความรู้รอบตัวเรื่องราศีกับผู้เล่นอีกด้วย

ปัญหาที่ผู้คนเผชิญหน้าก่อนที่จะมีโปรแกรมนี้

1. ก่อนที่จะมีโปรแกรม สมาชิกภายในกลุ่มไม่ค่อยเห็นแนวทางในการทำโปรเจ็กต์ที่หลากหลายและง่ายต่อการจัดทำ
2. การเปลี่ยนหัวข้อกระทันหันทำให้ เกิดความไม่เข้าใจในการทำงานของพวกเรา

อะไรคือแรงจูงใจที่ทำให้คุณสนใจหรือเลือกทำงานในโครงการนี้ (แอ็พพลิเคชันนี้)

จากปัญหาเบื้องต้นทำให้สมาชิกในกลุ่มได้ศึกษาหาเกมที่มีโครงสร้างไม่ซับซ้อนจากทางอินเทอร์เน็ต แล้วได้พบกับเกมเกมหนึ่ง ที่อาจทำได้ และไม่ยุ่งยากจนเกินไป จึงเลือกที่จะทำเกมเรียงไพ่เพื่อให้ง่ายต่อการจัดทำ และเกมเรียงไพ่นี้จัดอยู่ในรูปแบบที่สามารถเล่นได้ในทุกเพศทุกวัย

อธิบายโครงสร้างข้อมูล (Data Structure) ต่างๆ หรือ วิธีการต่างๆ ที่สามารถนำไปใช้สร้างแอ็พพลิเคชันนี้ได้

ในส่วนของ Data Structure ที่ใช้คือ Stack ซึ่งใช้หลักการ Last In First Out โดยการใช้ Stack เก็บข้อมูลของการ์ดแต่ละใบโดยให้เก็บไว้ในตัวแปร deck และใช้ Stack ในการเก็บไพ่ส่วนของ Waste ส่วนที่เปิดไพ่บนสุดของกองออกมาจะถูกเก็บไว้ที่ตัวแปรนี้ด้วยคำสั่ง deck.pop() เป็นต้น

วิธีการดำเนินงาน

ประเภทโครงสร้างข้อมูล (Data Structure) ที่ใช้

Stack และ Array

อธิบายการทำงานของโครงสร้างข้อมูลที่ใช้อย่างสั้น ๆ เช่น Queue ทำงานอย่างไร มี method อะไรบ้าง

Stack ทำงานโดยใช้หลักการ LastInFirstOut(LIFO) คือ เมื่อใส่ข้อมูลเข้าใน stack ตัวสุดท้ายจะออกมาตัวแรกสุด

Method

-push() ใส่ข้อมูลลงใน stack

-pop() เข้าข้อมูลออกมาจาก stack

-top() จะเป็นการเช็คค่าตัวที่อยู่ตัวสุดท้ายของ stack

-size() เช็คว่ามีข้อมูลใน stack เท่าไร

โดยโปรแกรมนี้ได้ใช้ Method ต่างๆดังนี้

-ใช้ push() ในการนำการ์ดที่เหลือจากการแจกไปยังแถวต่างๆเข้าไปในสำรับ

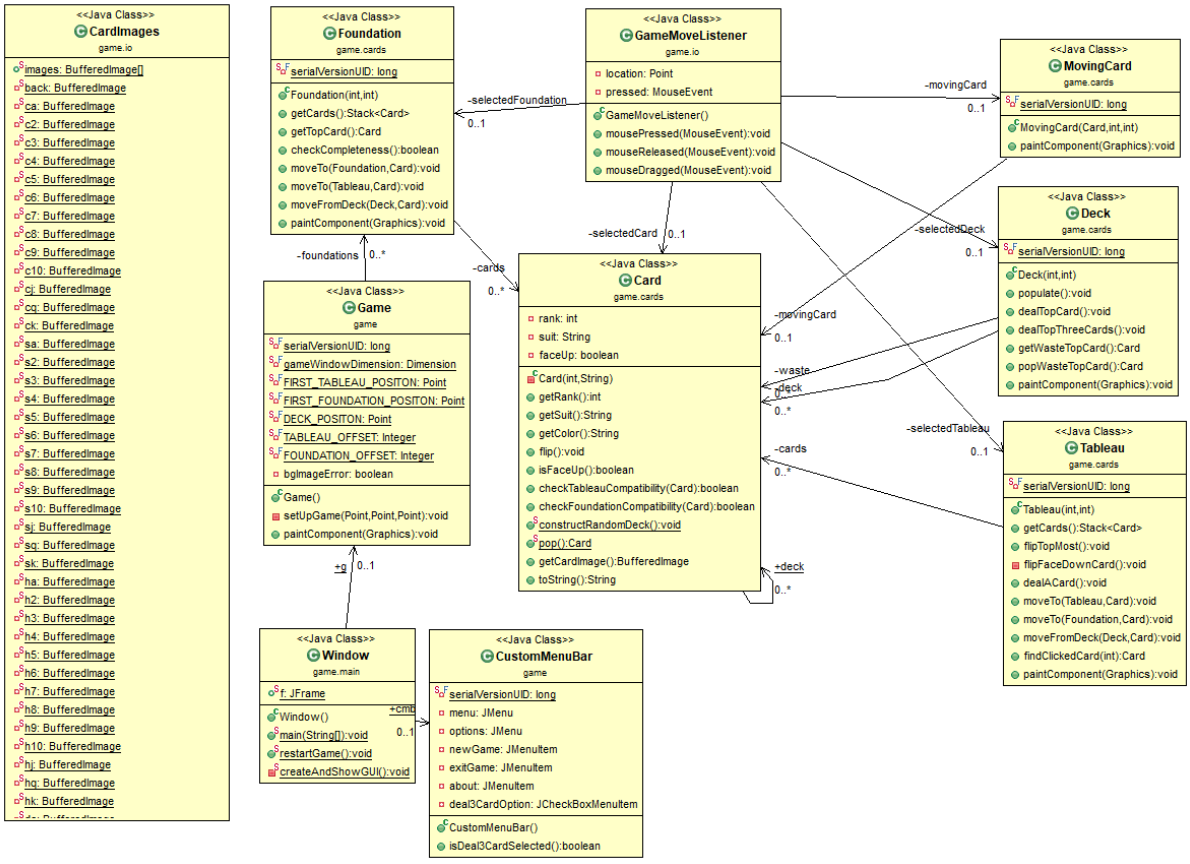
-ใช้ pop() เมื่อมีการคลิกที่กองการ์ด จะทำการ pop() ไพ่ใบบนสุดของสำรับออกมาข้างๆกองการ์ด

อธิบายว่าแอปพลิเคชันของคุณใช้โครงสร้างข้อมูลอย่างไร

ทำโดยการนำข้อมูลของการ์ดทั้งสำรับ push เข้าไปใน stack หากมีการคลิกที่สำรับจะทำการ pop การ์ดออกมา

อธิบายขั้นตอนวิธี (Algorithm)

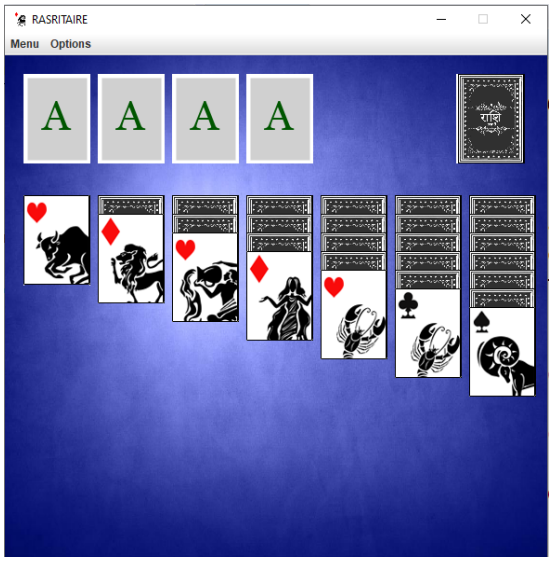
1. เริ่มจากการเพิ่มการ์ดทั้งหมดเก็บเข้าไว้ใน deck ก่อนจากนั้น เรียกใช้ Method shuffle เพื่อทำการสลับไพ่ในกองการ์ด
2. จากนั้นจะทำการแจกไพ่ในสำรับลงมาในแต่ละแถว และการ์ดที่เหลือจะเก็บไว้ที่ deck อย่างเดิม
3. ในแต่ละแถวที่แจกออกมาจะมีการตรวจสอบว่า ถ้าการ์ดในแถวยังไม่หมดและการ์ดใบล่างสุดไม่ได้หงายอยู่ ให้ทำการหงายไพ่ใบล่างสุดนั้น
4. ทำการเรียงไพ่แต่ละใบตามที่ระบุไว้ข้างต้นโดยที่ไพ่สีดำต้องเรียงสลับกับไพ่สีแดง
5. ไพ่ราศีมังกร สามารถนำมาอยู่เหนือไพ่ทั้ง 7 กองได้
6. สี่กองข้างบนต้องเริ่มจากไพ่ราศีมังกร ถ้าเจอไพ่ราศีมังกรและแยกเอาไปวางไว้ใน 4 ช่องข้างบนแล้ว (สุดท้ายต้องมีไพ่ราศีมังกรข้างบน 4 ใบ) ก็ให้หยิบไพ่ที่ดอกตรงกันกับไพ่ราศีมังกรข้างบนซ้อนทับต่อไปเรื่อยๆ ตามจำนวนที่เพิ่มขึ้น (ไล่จาก ราศีมังกร ไป ราศีกุมภ์ ราศีมีน ราศีเมษ ราศีพฤษภ ราศีเมถุน ราศีกรกฎ ราศีสิงห์ ราศีกันย์ ราศีตุล ราศีพิจิก ราศีธนู จนถึง ราศีคนแบกงู)
7. หากจำนวนไพ่ที่หงายอยู่ไม่สามารถนำมาเรียงต่อกันได้ ก็สามารถคลิกที่กองการ์ดเพื่อ pop ใบบนสุดออกมาจากกองการ์ด เพื่อนำการ์ดใบนั้นมาเล่นกับไพ่ในแต่ละแถวได้
8. ถ้ามีไพ่ใบไหนซ้อนอยู่ข้างหลัง ให้โยกย้ายไพ่ใบข้างหน้าไปมาจนกว่าจะหยิบใบที่ต้องการออกมาได้ แล้วเอาไปเรียงตามตำแหน่งต่อไป
9. ถ้าหยิบไพ่จากกองใดกองหนึ่งใน 7 กองขึ้นมาจนครบ ก็จะเกิดช่องว่างใช้วางไพ่ราศีคนแบกงูได้ (และเฉพาะไพ่ราศีคนแบกงูเดียวเท่านั้น)
10. เกมจะจบก็ต่อเมื่อจำนวนไพ่ 4 กองด้านบนเรียงจากไพ่ราศีมังกร(ล่างสุด) จนถึง ไพ่ราศีคนแบกงู(บนสุด)



ภาพที่1 : class diagram

ผลการดำเนินงาน

ภาพหน้าจอ (Screenshot)



ภาพที่2 : หน้าของตัวเกม

ภาพที่3 : เมื่อเล่นเกมจบ

วิเคราะห์ความเร็วหรือความซับซ้อนของโปรแกรม (โดยใช้ Big O และ การอธิบายประกอบ)

```
private void setUpGame(final Point tableauPos, final Point foundationPos, final Point deckPos) {
    Card.constructRandomDeck();

    final Tableau[] tableau = new Tableau[7];
    for (int tableauIndex = 1; tableauIndex <= tableau.length; tableauIndex++) {
        tableau[tableauIndex - 1] = new Tableau((int) tableauPos.getX() + Game.TABLEAU_OFFSET * (tableauIndex - 1), (int) tableauPos.getY());
        for (int numberOfCards = 0; numberOfCards < tableauIndex; numberOfCards++) {
            tableau[tableauIndex - 1].dealACard();
        }
        tableau[tableauIndex - 1].flipTopMost();
        super.add(tableau[tableauIndex - 1]);
    }

    foundations = new Foundation[4];
    for (int i = 0; i < foundations.length; i++) {
        foundations[i] = new Foundation((int) foundationPos.getX() + Game.FOUNDATION_OFFSET * i, (int) foundationPos.getY());
        super.add(foundations[i]);
    }
```

ภาพที่4 : เมธอด setupGame ของคลาส Game.java

```
for (int i = 0; i < foundations.length; i++) {
    if (!foundations[i].checkCompleteness()) {
        isGameOver = false;
    }
}
```

ภาพที่5 : ของคลาส Game.java

```
public void populate() {
    while (!Card.deck.isEmpty()) {
        deck.push(Card.pop());
    }
}
```

ภาพที่6 : เมธอด Populate ของคลาส Deck.java

```java
public void dealTopThreeCards() {
    Card topMost = null;
    Card secondMost = null;
    Card thirdMost = null;
    if (!deck.isEmpty()) {
        topMost = deck.pop();
        topMost.flip();
        waste.push(topMost);
        if (!deck.isEmpty()) {
            secondMost = deck.pop();
            secondMost.flip();
            waste.push(secondMost);
        }
        if (!deck.isEmpty()) {
            thirdMost = deck.pop();
            thirdMost.flip();
            waste.push(thirdMost);
        }
    } else {
        while (!waste.isEmpty()) {
            deck.push(waste.pop());
            deck.peek().flip();
        }
    }
}
```

ภาพที่7 : เมธอด dealTopThreeCards ของคลาส Deck.java

```java
public void dealTopCard() {
    if (!deck.isEmpty()) {
        waste.push(deck.pop());
        waste.peek().flip();
    } else {
        while (!waste.isEmpty()) {
            deck.push(waste.pop());
            deck.peek().flip();
        }
    }
}
```

ภาพที่8 : เมธอด dealTopCard ของคลาส Deck.java

```java
public void moveTo(final Tableau destination, final Card c) {
    final Stack<Card> destinationStack = destination.cards;
    if (!destinationStack.isEmpty()) {
        if (destinationStack.peek().checkTableauCompatibility(c)) {
            final Deque<Card> toBeMovedCards = new ArrayDeque<>();
            while (!cards.isEmpty()) {
                final Card tmp = cards.pop();
                toBeMovedCards.push(tmp);
                if (tmp.equals(c)) {
                    break;
                }
            }
            while (!toBeMovedCards.isEmpty()) {
                destinationStack.push(toBeMovedCards.pop());
            }
        }
    } else {
        if (c.getRank() == 13) {
            final Deque<Card> toBeMovedCards = new ArrayDeque<>();
            while (!cards.isEmpty()) {
                final Card tmp = cards.pop();
                toBeMovedCards.push(tmp);
                if (tmp.equals(c)) {
                    break;
                }
            }
            while (!toBeMovedCards.isEmpty()) {
                destinationStack.push(toBeMovedCards.pop());
            }
        }
    }
    flipFaceDownCard();
}
```

ภาพที่9 : เมธอด Moveto ของคลาส Tableau.java

```java
for (final Card c : handler) {
    g.drawImage(c.getCardImage(), 0, cardYPos, 72, 96, this);
    cardYPos += 20;
}
```

ภาพที่10 : ของคลาส Tableau.java

จากภาพที่ 5 จะได้ จะได้ Big-O เท่ากับ **O(n)**

จากภาพที่ 4,7,8 และ 9 จะได้ Big-O เท่ากับ **O(n log n)**

จากภาพที่ 6 และ 10 จะได้ Big-O เท่ากับ **O(log n)**

ส่วนเมธอดอื่นๆส่วนมากก็จะมี Big-O = **0(1)**

จึงสรุปได้ว่าตัวโปรแกรมของเรานั้นมีความเร็วอยู่ในระดับที่ ปานกลางค่อนไปทางเร็ว

เนื่องจากความซับซ้อนของลูปนั้นมีอยู่แค่ไม่กี่เมธอดนั่นเอง

สรุปผล

อธิบายความรู้ความเข้าใจใหม่ ๆ จากการทำงานในโครงการนี้

สิ่งใหม่ที่ได้จาก การทำงานในครั้งนี้ก็คือ การใช้ Data Structure Stack มาใช้ทำโปรเจค และ การใช้ BufferedImage ในการนำภาพมาใส่ในโปรแกรมนั่นเอง

อุปสรรค / ปัญหาระหว่างโครงการและวิธีแก้ปัญหา

เนื่องจากสถานะการณ์ปัจจุบันของการระบาดโควิด-19 ทำให้การพบกันของสมาชิกในกลุ่มลดลงและขาดความเข้าใจในการทำงานร่วมกัน

ทางแก้ไขปัญหา : ปรับตัวมาคุยการออนไลน์ผ่านโปรแกรม Discord และ Google Meet

เนื่องจากตัวโปรเจคนั้นทางเราไม่ได้เขียนขึ้นมาเองทำให้ไม่รู้ว่าโค้ดส่วนไหนใช้ทำอะไรนั่นเอง

ทางแก้ไขปัญหา : ทำการศึกษาตัวโปรแกรมควบคู่กับหาข้อมูลในอินเตอร์เน็ตเพื่อทำความเข้าใจในตัวโค้ด

การปรับปรุงโปรแกรมประยุกต์นี้เป็นไปได้ในอนาคต

เนื่องจากโปรแกรมนี้ฟังก์ชันที่มีอยู่น้อย เข้าใจง่าย ไม่ซับซ้อน พวกเราคิดว่าในอนาคตสามารถสร้างฟังก์ชันอื่นๆ ขึ้นมาเพิ่มได้ อาทิเช่น มีการจับเวลา มีปุ่มHint มีการรับข้อมูลผู้เล่น เป็นต้น

เอกสารอ้างอิง

aeris170. (2563). Solitaire. [เว็บบล็อก]. สืบค้นจาก https://github.com/aeris170/Solitaire

Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, Data Structures and Algorithms in Java,6th Edition, Wiley, 2014