Homework 3 CS111

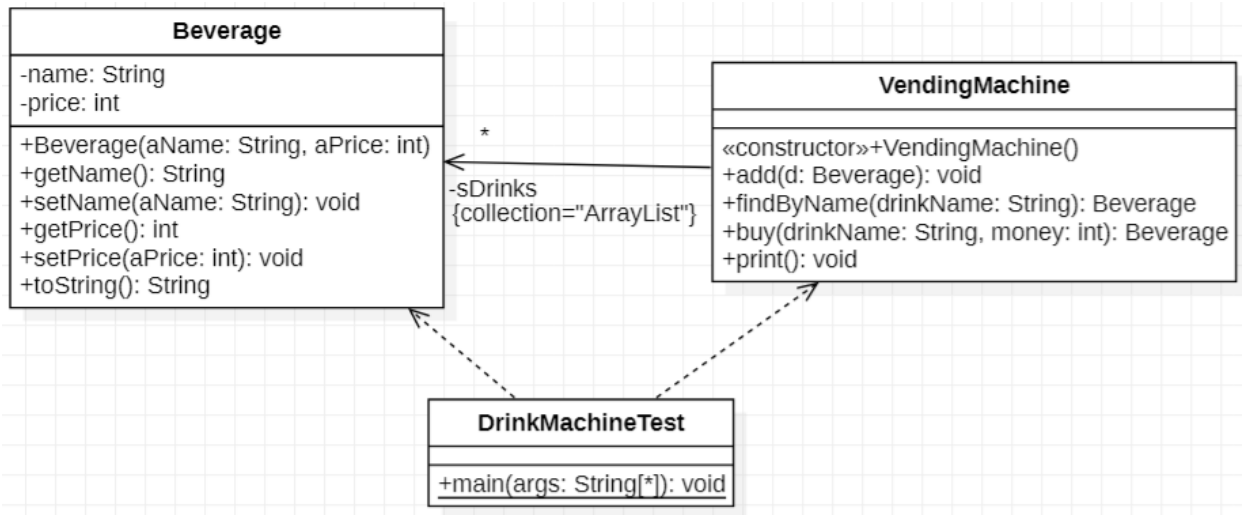
กำหนดส่ง 17 เมษายน 2566

<u>จุดมุ่งหมาย</u>

- เพื่อให้สามารถเขียนโค้ดสำหรับคลาสตามแผนภาพที่ให้
- เพื่อให้สามารถใช้อาร์เรย์หนึ่งและสองมิติในการแก้ปัญหาได้
- เพื่อให้สามารถใช้อาร์เรย์ลิสต์ในการแก้ปัญหาได้

<u>คำสั่ง</u>

1. ให้เขียนโค้ดเพื่อจำลองเครื่องขายน้ำหยอดเงิน โดยโปรแกรมนี้ประกอบด้วยคลาสดังไดอะแกรมที่ให้



<u>คำสั่ง</u>

1.1. ให้เขียนคลาส **`Beverage`** ดังไดอะแกรม ภายในคลาสนี้มี 2 instance variables, **`name`** และ **`price`** พร้อมเมท็อด constructor, getter และ setter ของตัวแปรเหล่านี้ เมท็อด **`toString`** คืนค่า String ที่สามารถใช้ในการพิมพ์วัตถุ Beverage โดยคืนอยู่ในรูปของ

```
name [price %d]
```

ตัวอย่างเช่น วัตถุ Beverage ที่มีชื่อว่า Coke Zero และมีราคาเป็น 17 บาท ถูกคืนเป็นข้อความ

```
Coke Zero [price 17]
```

Note ถ้าต้องการคืนค่า String โดยทำ format ได้ (คล้ายคำสั่ง printf) ให้ใช้เมท็อด `String.format`

1.2. ให้เขียนคลาส **`VendingMachine`** ที่แทนเครื่องขายเครื่องดื่มอัตโนมัติ โดย Beverage และ VendingMachine มีความสัมพันธ์และรายละเอียดตามภาพไดอะแกรมที่ให้

คลาส **`VendingMachine`** ภายในมีตัวแปรวัตถุชนิด ArrayList ของ `Beverage` (**`sDrinks`**) และมีเมท็อดดังนี้

- เมท็อด *constructor*: สร้างวัตถุ ArrayList เพื่อให้เป็นค่าเริ่มต้นกับตัวแปร **`sDrinks`**
- **`void add:`** เพิ่ม Beverage ใหม่ตามที่ได้รับจากพารามิเตอร์เข้าในตัวแปร **`sDrinks`**

- **`Beverage findByName:`** คืนวัตถุ Beverage ที่มีชื่อตรงกับพารามิเตอร์ drinkName ที่ส่งมายังเมท็อด โดยไม่คำนึงว่าชื่อเป็นตัวพิมพ์ใหญ่หรือเล็ก ในกรณีที่ไม่พบคืน null
- **`Beverage buy:`** ถ้าพบ beverage ที่มีชื่อตรงกับพารามิเตอร์ drinkName โดยใช้เมท็อด **`findByName`** และให้เงินมาเพียงพอกับราคาของ Beverage ให้ vending machine คืนวัตถุ Beverage ที่ต้องการซื้อนั้น แต่ถ้าไม่พบ หรือให้เงินไม่เพียงพอให้แสดงข้อความผิดพลาดและคืนค่า null
- **`void print:`** พิมพ์ทุก beverage ที่มีอยู่ในเครื่อง vending machine

1.3. เมื่อสร้างคลาสข้างต้นเสร็จแล้ว ให้เขียน**คลาสทดสอบ**เพื่อทดสอบการทำงาน โดยให้เพิ่มอย่างน้อย 5 Beverages ลงเครื่องขายน้ำ แล้วจึงเริ่มถามผู้ใช้ว่าต้องการซื้อน้ำชื่ออะไร มีเงินเท่าไร และแสดงผลที่ได้จากการซื้อน้ำ ให้วนถามไปเรื่อย ๆ จนกว่าผู้ใช้จะไม่ต้องการซื้ออีกแล้ว

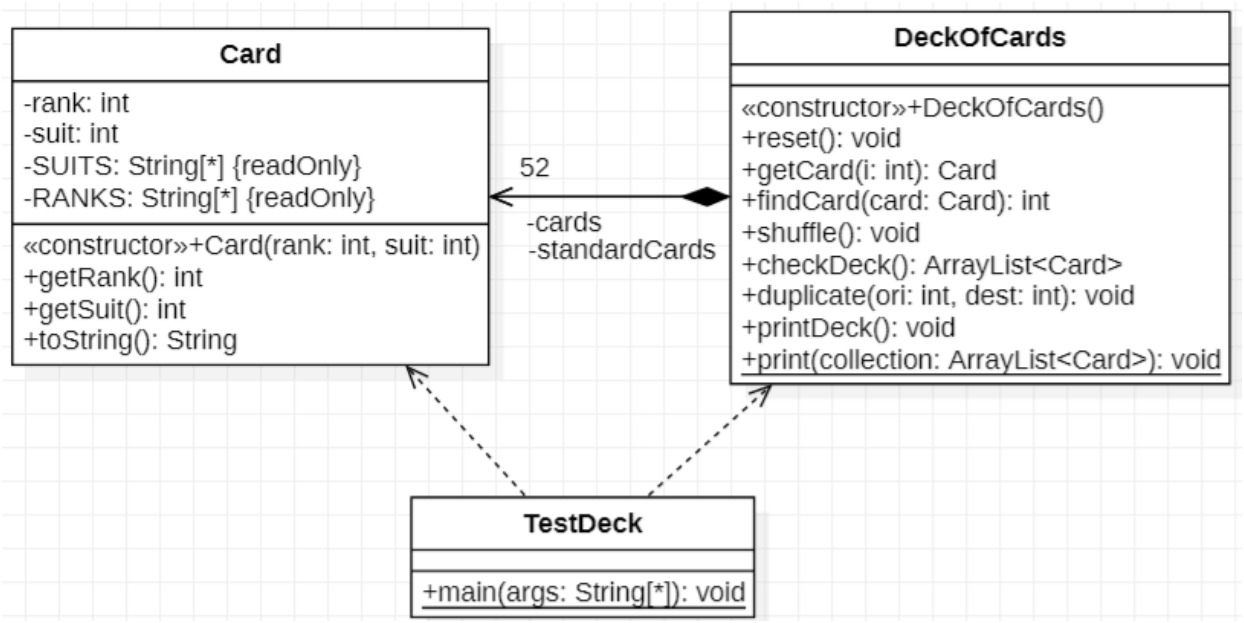
ตัวอย่างการทำงานที่ได้

```
Machine has:
 - Coke Zero [price 17]
 - Pepsi Max [price 16]
 - Fanta Strawberry [price 15]
 - Fanta Orange [price 15]
 - Coffee [price 45]
 - Expresso [price 60]
 - Green Tea [price 50]
What would you like to buy? Coke
How much do you have? 20
The machine did not serve this beverage Coke
Buy more? (y/n) y
Machine has:
 - Coke Zero [price 17]
 - Pepsi Max [price 16]
 - Fanta Strawberry [price 15]
 - Fanta Orange [price 15]
 - Coffee [price 45]
 - Expresso [price 60]
 - Green Tea [price 50]
What would you like to buy? Fanta strawberry
How much do you have? 50
Here is your drink: Fanta Strawberry [price 15]
Buy more? (y/n) y
Machine has:
 - Coke Zero [price 17]
 - Pepsi Max [price 16]
 - Fanta Strawberry [price 15]
 - Fanta Orange [price 15]
 - Coffee [price 45]
 - Expresso [price 60]
 - Green Tea [price 50]
What would you like to buy? expresso
How much do you have? 50
You gave 50 bahts. It's not enough money to get Expresso [price 60]
Buy more? (y/n) n
Bye
```

2. ให้คลาส Card ต่อไปนี้ (ให้ใช้คลาส Card ตามข้อกำหนดที่ให้ โดย**ไม่**เพิ่มเติมเมท็อดหรือตัวแปรใด ๆ)

```
public class Card {
	private int rank, suit;

	private final String[] SUITS = {"Club", "Diamond", "Heart", "Spade"};
	private final String[] RANKS = {"2", "3", "4", "5", "6", "7", "8",
			"9", "10", "Jack", "Queen", "King", "Ace"};

	public Card(int rank, int suit) {
		this.rank = rank;
		this.suit = suit;
	}

	public int getRank() {
		return rank;
	}

	public int getSuit() {
		return suit;
	}

	public String toString() {
		return RANKS[rank] + " of " + SUITS[suit];
	}

	public boolean equals(Card other) {
		return this.rank == other.rank && this.suit == other.suit;
	}
}
```

ให้สร้างคลาส **DeckOfCards** เพื่อแทนสำรับไพ่มาตรฐาน 52 ใบ ซึ่งประกอบไปด้วย 4 suits ได้แก่ โพธิ์แดง (hearts), ข้าวหลามตัด (diamonds), โพธิ์ดำ (spades) และดอกจิก (clubs) และในแต่ละ suit มีไพ่ 13 หมายเลข คือ 2, 3, ..,10, J, Q, K และ Ace ในคลาส **DeckOfCards** มี instance variable, **standardCards** และ **cards** ซึ่งเป็นอาร์เรย์หนึ่งมิติขนาด 52 สำหรับเก็บ **Card**



a) เมท็อด *constructor* ทำหน้าที่สร้างสำรับไพ่มาตรฐานโดยสร้างวัตถุอาร์เรย์ของ Card ขนาด 52 สำหรับ **standardCards** และ **cards** หลังจากนั้นสร้างไพ่ที่เป็นไปได้ทั้ง 52 ใบของทั้งสำรับสำหรับ **standardCards** กล่าวอีกนัยคือสร้างไพ่แล้วกำหนดให้เป็นสมาชิกของ standardCards นั่นเอง หลังจากนั้นให้สำเนา standardCards ไปให้ cards โดยที่ทั้ง 2 สำรับจะเป็นอาร์เรย์คนละชุดกัน (มีค่าเหมือนกันแต่ใช้อาร์เรย์คนละตัวกัน) ซึ่งการเปลี่ยนแปลงตำแหน่งหรือไพ่ของชุดหนึ่งต้องไม่กระทบไพ่อีกสำรับหนึ่ง

Note การสำเนาอาร์เรย์ทั้งสองตัวนี้เป็นการสำเนา **ไม่ใช่**การกำหนดให้มันไปอ้างถึงวัตถุอาร์เรย์ตัวเดียวกัน นั่นคือ ถ้าไพ่ใน cards เปลี่ยนตำแหน่งต้องไม่ส่งผลกับ standardCards

b) **void reset()** ทำหน้าที่ reset ให้สำรับไพ่ **cards** มีชุดไพ่แบบเดียวกับ **standardCards**

Hint อาจใช้ System.arraycopy ช่วยในการทำงาน

c) **Card getCard(int i)** คืนค่า Card ในสำรับ **cards** ตามตำแหน่งที่ระบุให้ในพารามิเตอร์

d) **int findCard(Card card)** ค้นหาตำแหน่งของไพ่ในสำรับ **cards** ตามพารามิเตอร์ card ที่ต้องการค้นหา คืนตำแหน่งที่พบตัวแรก และคืน -1 ถ้าไม่พบไพ่นั้น

e) **void shuffle()** ทำหน้าที่สลับไพ่ในสำรับ **cards** โดยการสลับไพ่แบบสุ่ม

Hint: การสลับทำโดยสุ่มเลือก 2 ตำแหน่งในสำรับ แล้วสลับไพ่ที่อยู่ในตำแหน่งเหล่านั้น ให้ทำวนซ้ำอย่างน้อย 15 รอบ ไพ่ทั้งสำรับก็จะเหมือนกับถูกสับไพ่เวลานำไปเล่น (ในการสุ่มเลข ไม่จำเป็นต้องตรวจสอบว่าเลขตำแหน่งที่ได้เป็นเลขซ้ำกันหรือไม่)

f) **ArrayList<Card> checkDeck()** ทำหน้าที่ตรวจสอบเทียบกับ **standardCards** ว่าในสำรับไพ่ **cards** มีไพ่ใบใดขาดไปบ้าง กรณีที่มีไพ่ครบทุกใบให้คืน null ในกรณีที่ไพ่ใบใดหายไป ให้รวบรวมไพ่ที่หายไปเหล่านั้นเก็บใน ArrayList<Card> เพื่อเป็นค่าส่งคืนจากเมท็อดนี้

Hint ให้วนทำซ้ำ ไพ่แต่ละใบใน **standardCards** โดยใช้เมท็อด **findCard** ช่วยในการทำงาน

g) **void duplicate(int ori, int dest)** ทำสำเนาไพ่ในตำแหน่งต้นทาง **ori** ตามพารามิเตอร์ที่ได้รับไปแทนไพ่ในตำแหน่งปลายทาง **dest** ผลการทำงานจะทำให้ไพ่ในตำแหน่งปลายทางเหมือนกับไพ่ในตำแหน่งต้นทาง ให้ตรวจสอบด้วยว่าตำแหน่งต้นทางและปลายทางเป็นตำแหน่งที่เป็นไปได้ในสำรับไพ่ **cards** ในกรณีที่ตำแหน่งไม่ถูกต้อง ไม่ต้องดำเนินการใดๆ

h) **void printDeck()** พิมพ์ไพ่ทุกใบในสำรับ (ที่เก็บอยู่ในตัวแปรวัตถุ **cards**) ออกทางหน้าจอ

i) **void print(ArrayList<Card collection)** เป็น static method ที่รับพารามิเตอร์เป็น ArrayList<Card> เพื่อพิมพ์ไพ่ที่มีใน collection ออกทางหน้าจอ ในกรณีที่ collection เป็น null จะพิมพ์ข้อความแสดงว่าไม่มีอะไรให้พิมพ์ (ดังตัวอย่างผลการทำงาน)

โค้ดที่ให้ต่อไปนี้เป็นคลาสทดสอบ และตัวอย่างผลลัพธ์จากการทำงานที่ได้

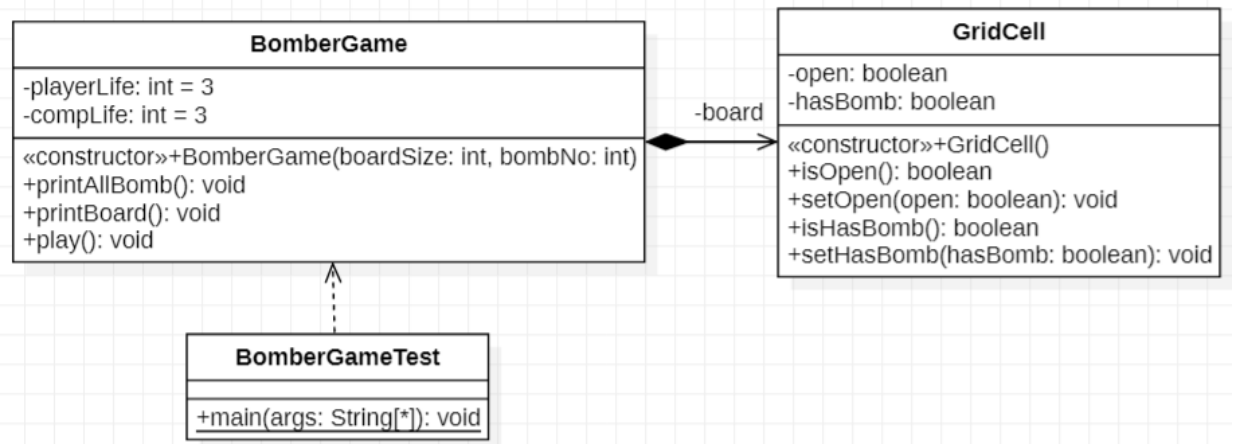
```java
public class TestDeck {

	public static void main(String[] args) {
		DeckOfCards deck = new DeckOfCards();
		deck.printDeck();

		Card c = new Card(1, 0); // 3 of Club
		System.out.printf("\t>>>%s is in the deck's position %d\n", c,
deck.findCard(c));
		System.out.println("\n*******After shuffled********");
		deck.shuffle();

		System.out.println("After shuffle, missing cards from the deck: ");
		DeckOfCards.print(deck.checkDeck());
		deck.printDeck();
		System.out.printf("\t>>>%s is in the deck's position %d\n", c,
deck.findCard(c));

		deck.reset();
		deck.duplicate(4, 10);
		deck.duplicate(0, 51);
		deck.duplicate(0, 52);
		System.out.println("\n*******After duplicate, missing cards from the
deck: ");
		DeckOfCards.print(deck.checkDeck());
		deck.printDeck();
	}
}
```

Sample result:

```
Deck of Cards (52 cards):
2 of Club      2 of Diamond   2 of Heart     2 of Spade
3 of Club      3 of Diamond   3 of Heart     3 of Spade
4 of Club      4 of Diamond   4 of Heart     4 of Spade
5 of Club      5 of Diamond   5 of Heart     5 of Spade
6 of Club      6 of Diamond   6 of Heart     6 of Spade
7 of Club      7 of Diamond   7 of Heart     7 of Spade
8 of Club      8 of Diamond   8 of Heart     8 of Spade
9 of Club      9 of Diamond   9 of Heart     9 of Spade
10 of Club     10 of Diamond  10 of Heart    10 of Spade
Jack of Club   Jack of Diamond       Jack of Heart  Jack of Spade
Queen of Club  Queen of Diamond      Queen of Heart Queen of Spade
King of Club   King of Diamond       King of Heart  King of Spade
Ace of Club    Ace of Diamond Ace of Heart   Ace of Spade
       >>>3 of Club is in the deck's position 4

*******After shuffled********
After shuffle, missing cards from the deck:
null collection. Nothing to print!
Deck of Cards (52 cards):
2 of Club      2 of Diamond   2 of Heart     2 of Spade
9 of Spade     3 of Diamond   Queen of Diamond      3 of Spade
3 of Heart     4 of Diamond   9 of Heart     4 of Spade
5 of Club      5 of Diamond   5 of Heart     5 of Spade
6 of Club      King of Spade  6 of Spade     Jack of Diamond
7 of Club      7 of Diamond   10 of Club     10 of Diamond
9 of Diamond   8 of Diamond   8 of Heart     Ace of Diamond
9 of Club      8 of Club      4 of Heart     3 of Club
7 of Heart     7 of Spade     10 of Heart    King of Diamond
Jack of Club   6 of Heart     Jack of Heart  Jack of Spade
Queen of Club  4 of Club      8 of Spade     Queen of Spade
King of Club   10 of Spade    King of Heart  6 of Diamond
```

```
Ace of Club    Queen of Heart Ace of Heart   Ace of Spade
       >>>3 of Club is in the deck's position 31

*******After duplicate, missing cards from the deck:
Collection contains:
4 of Heart     Ace of Spade
Total 2 cards
Deck of Cards (52 cards):
2 of Club      2 of Diamond   2 of Heart     2 of Spade
3 of Club      3 of Diamond   3 of Heart     3 of Spade
4 of Club      4 of Diamond   3 of Club      4 of Spade
5 of Club      5 of Diamond   5 of Heart     5 of Spade
6 of Club      6 of Diamond   6 of Heart     6 of Spade
7 of Club      7 of Diamond   7 of Heart     7 of Spade
8 of Club      8 of Diamond   8 of Heart     8 of Spade
9 of Club      9 of Diamond   9 of Heart     9 of Spade
10 of Club     10 of Diamond  10 of Heart    10 of Spade
Jack of Club   Jack of Diamond       Jack of Heart  Jack of Spade
Queen of Club  Queen of Diamond      Queen of Heart Queen of Spade
King of Club   King of Diamond       King of Heart  King of Spade
Ace of Club    Ace of Diamond Ace of Heart   2 of Club
```

3. ให้สร้างเกมเหยียบระเบิดซึ่งผู้เล่นแข่งกับคอมพิวเตอร์เปิดช่องบนกระดาน ผู้เล่นและคอมพิวเตอร์มีชีวิตเริ่มต้น 3 ชีวิต ถ้าเปิดเจอระเบิดพลังชีวิตจะลดไปทีละ 1 เกมจบเมื่อมีฝ่ายหนึ่งตาย (พลังชีวิตเหลือ 0) คลาสสำหรับเกมนี้มีคุณสมบัติดังแสดงต่อไปนี้



`GridCell` เป็นคลาสเก็บข้อมูลของช่องหนึ่งช่องบนกระดาน โดยมีตัวแปรวัตถุคือ `open` บอกสถานะว่าช่องนั้นถูกเปิดหรือยัง และ `hasBomb` บอกว่าช่องนั้นมีระเบิดหรือไม่ คลาสนี้มีเมท็อดคือ getter (getter ของค่าบูลีนนิยมตั้งชื่อขึ้นต้นด้วยคำว่า is แทนคำว่า get) และ setter ของทั้งสองคุณสมบัติ และ เมท็อดตัวสร้างตั้งค่า `open` และ `hasBomb` เป็นค่าเท็จ

`BomberGame` เป็นคลาสตัวแทนเกมซึ่งมีตัวแปรวัตถุ `playerLife` และ `comLife` สำหรับเก็บพลังชีวิตของผู้เล่นและคอมพิวเตอร์ตามลำดับ นอกจากนี้ คลาสนี้ยังมีตัวแปรวัตถุ `board` เพื่อเป็นตัวแทนของกระดาน `board` ที่เป็นชนิด**อาร์เรย์สองมิติ**ของ `GridCell` คลาสนี้มีเมท็อดดังนี้

- เมท็อด *constructor* รับค่าขนาดของกระดาน (`boardSize`) และ จำนวนระเบิด (`bombNo`) เมท็อดนี้สร้างกระดาน ซึ่งมีจำนวนแถวและคอลัมน์เท่าที่กำหนดในพารามิเตอร์ `boardSize` แต่ถ้า `boardSize` ที่

กำหนดมีค่าน้อยกว่า 3 ให้สร้างกระดานขนาด 3 x 3 และ วางระเบิดลงในช่องกระดานอย่างสุ่มตามจำนวน bombNo ที่กำหนด ถ้า bombNo ที่กำหนดมีค่าน้อยกว่า 5 หรือมากกว่าจำนวนเซลทั้งหมดของกระดานให้วาง 5 ลูก

- **void printAllBomb():** พิมพ์กระดานเพื่อแสดงระเบิดทั้งหมดโดยไม่สนใจว่าช่องปิดหรือเปิดอยู่ ให้พิมพ์ช่องที่ไม่มีระเบิดด้วย '-' และ ช่องที่มีระเบิดด้วย 'b' ดังแสดงในตัวอย่างข้างล่าง
- **void printBoard():** พิมพ์กระดานเพื่อแสดงสถานะของกระดาน โดยถ้าช่องปิดอยู่ ให้แทนช่องนั้นด้วย 'X' ถ้าช่องเปิดและไม่มีระเบิดให้แทนด้วย '-' และ ช่องเปิดที่มีระเบิดแทนด้วย 'b' ดังแสดงในตัวอย่างข้างล่าง
- **void play():** เริ่มต้นให้สุ่มเลือกผู้เริ่มเล่นก่อน แล้วผลัดกันระหว่างผู้เล่นและคอมพิวเตอร์ในการเลือกช่องที่จะเปิด สำหรับผู้เล่นให้ถามตำแหน่งแถวและคอลัมน์ของช่อง สำหรับคอมพิวเตอร์ให้เลือกช่องอย่างสุ่ม ถ้าช่องที่เลือกไม่มีอยู่จริง (index ผิด) หรือเปิดไปแล้ว ให้เลือกใหม่จนกว่าจะได้ช่องที่ปิด เมื่อเปิดช่องถ้าช่องมีระเบิด ลดพลังชีวิตคนที่เปิดลง 1 จากนั้นให้พิมพ์สถานะของกระดาน (ใช้ printBoard) และผู้เล่นทั้งหมด เลิกเล่นเมื่อพลังชีวิตของฝ่ายใดฝ่ายหนึ่งเหลือ 0 เมื่อจบเกม ให้แสดงตำแหน่งระเบิดทั้งหมดในกระดาน (ใช้ printAllBomb)

ตัวอย่างการใช้ BomberGame เป็นดังต่อไปนี้

```
public class BomberGameTest {
	public static void main(String[] args) {
		BomberGame bg = new BomberGame(4, 2); //Get 4x4 board and 5 bombs
		bg.play();
	}
}
```

ตัวอย่างของการทำงานของโปรแกรมทดสอบเป็นดังนี้

```
------- Player Turn -------
Input row: 4
Input col: 5
Invalid row or col or cell is already opened. Choose again
Input row: 2
Input col: 1
Opening Grid 2, 1
X X X X
X X X X
X b X X
X X X X
Player Life: 2
Computer Life: 3
------- Computer Turn -------
Opening Grid 3, 1
X X X X
X X X X
X b X X
X b X X
Player Life: 2
Computer Life: 2
------- Player Turn -------
Input row: 0
Input col: 0
```

```
Opening Grid 0, 0
- X X X
X X X X
X b X X
X b X X
Player Life: 2
Computer Life: 2
------- Computer Turn -------
Opening Grid 2, 3
- X X X
X X X X
X b X b
X b X X
Player Life: 2
Computer Life: 1
------- Player Turn -------
Input row: 3
Input col: 1
Invalid row or col or cell is already opened. Choose again
Input row: 0
Input col: 2
Opening Grid 0, 2
- X - X
X X X X
X b X b
X b X X
Player Life: 2
Computer Life: 1
------- Computer Turn -------
Opening Grid 2, 0
- X - X
X X X X
- b X b
X b X X
Player Life: 2
Computer Life: 1
------- Player Turn -------
Input row: 0
Input col: 1
Opening Grid 0, 1
- - - X
X X X X
- b X b
X b X X
Player Life: 2
Computer Life: 1
------- Computer Turn -------
Opening Grid 1, 2
- - - X
X X - X
- b X b
X b X X
Player Life: 2
Computer Life: 1
------- Player Turn -------
Input row: 0
Input col: 3
Opening Grid 0, 3
- - - -
X X - X
- b X b
X b X X
Player Life: 2
Computer Life: 1
------- Computer Turn -------
Grid 0, 1 is already opened. Choose again
```

```
Opening Grid 3, 2
- - - -
X X - X
- b X b
X b - X
Player Life: 2
Computer Life: 1
------- Player Turn -------
Input row: 1
Input col: 1
Opening Grid 1, 1
- - - -
X - - X
- b X b
X b - X
Player Life: 2
Computer Life: 1
------- Computer Turn -------
Opening Grid 2, 2
- - - -
X - - X
- b - b
X b - X
Player Life: 2
Computer Life: 1
------- Player Turn -------
Input row: 1
Input col: 3
Opening Grid 1, 3
- - - -
X - - -
- b - b
X b - X
Player Life: 2
Computer Life: 1
------- Computer Turn -------
Grid 1, 3 is already opened. Choose again
Grid 2, 0 is already opened. Choose again
Opening Grid 3, 3
- - - -
X - - -
- b - b
X b - b
Player Life: 2
Computer Life: 0
You win!

------------- Game Ended. All Bombs' Location are -----------
- - - -
b - - -
- b - b
- b - b
```

หมายเหตุ คลาส BomberGame นอกจากเมท็อดที่กำหนดในไดอะแกรมแล้ว **อนุญาต**ให้สร้าง private method เพื่อช่วยแบ่งโค้ดให้อ่านขียน และจัดแบ่งการทำงานให้สะดวกและง่ายขึ้นได้