Project 2 - BINGO

CS 0401 — Intermediate Programming using Java

Check the Due Date on the CourseWeb

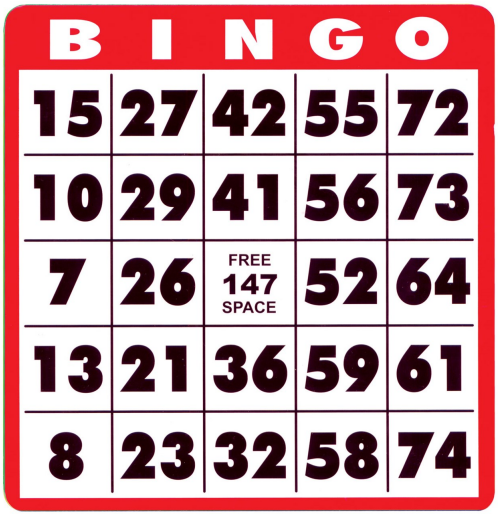
Bingo

(Modified from Wikipedia) In the United State, **Bingo** is a game of chance in which each player matches numbers printed in different arrangements on 5×5 cards which the numbers the game host (caller) draws at random, marking the selected numbers with tiles. When a player finds the selected numbers are arranged on their card in a row, they call out "Bingo!" to alert all participants to a winning card, which prompts the game host (or an associate assisting the host) to examine the card for verification of the win. Players compete against one another to be the first to have a winning arrangement for the price or jackpot. After a winner is declared, the players clear their number cards of the tiles and the game host begins a new round of play.

Bingo Cards

The most common **Bingo cards** are flat piece of cardboard or disposable paper which contain 25 squares arranged in five vertical column and five side to side rows. Each space in the grid contains a number.

A typical Bingo game utilizes the numbers 1 through 75. The five columns of the card are labeled B, I, N, G and O from left to right. The center space is usually marked "Free" or "Free Space", and is considered automatically filled. The range of printed numbers that can appear on the card is normally restricted by column, with the B column only containing numbers between 1 and 15 inclusive, the I column containing only 16 through 30, N containing 31 through 45, G containing 46 through 60, and O containing 61 through 75. Here is an example of a Bingo card:



The Goal

The goal of this project is for you to create a host program that manages the Bingo game. Each round of game, there can be a number of players. In a round of game, a player can have any number of Bingo cards. To make it interesting, each card will costs $1. Half of the cost of a card will go to the host, the other half will be placed into a pot. When the game begins, the host will draw a Bingo ball (a ball with a number between 1 and 75) at random from a Bingo Cage. Balls will be drawn one at a time until a player gets the BINGO. Whoever wins that round will get all the money in the pot.

As usual, you are going to implement your program according to the Object-Oriented Programming (OOP). Read descriptions in the next sections carefully. You may think that there are way to many classes and some of them are not necessary. Just stick with the description. We want you to practice dealing with multiple classes and objects. Each class will have its own set methods. However, if you need to create additional methods, those methods must be declared `private`. Again, all classes suggested in the next sections must be implemented with methods described in the next sections.

Part I: The `BingoNumber` Class (5 Points)

A Bingo number is a number displayed on a Bingo card. A Bingo number can be marked by a player. Whether a number is marked, a player will always see the number of the card.

So, from the above observations, the `BingoNumber` class should have the following:

- **Constructors**:
  - `public BingoNumber(int aNumber)`: This constructor constructs an object of type `BingoNumber` with the given number (`aNumber`).
- **Accessor Methods**:
  - `public int getNumber()` returns the integer representing the number of the `BingoNumber`. Note that even if the number is marked, it still returns its number as usual.
  - `public boolean isMarked()` returns `true` or `false` whether the number is marked.
  - `public String toString()` returns a two-digit string presentation of a `BingoNumber`. The output string depends on whether the number if marked. If the number is not marked yet, simply show the number. However, if the number is marked, the output string will simply be `XX`. Here is an example of a code snippet and expected output where the `mark()` and `toNumberString()` methods will be discussed next:

```
BingoNumber n1 = new BingoNumber(5);
BingoNumber n2 = new BingoNumber(12);
n2.mark();
System.out.println(n1);
System.out.println(n1.toNumberString());
System.out.println(n2);
System.out.println(n2.toNumberString());
```

Output:

```
 5
 5
XX
12
```

  - `public String toNumberString()` returns a two-digit string representation of a `BingoNumber` as if it has not been marked yet.

- **Mutator Method**:

  - `public void mark()` marks the `BingoNumber`. Note that once the number is marked. It cannot be unmarked because we are not going to supply a method that will let users unmark a Bingo number.

- **Instance Variables**: It is your turn to figure it out what would be the set of instance variables of this class.

Part II: The `BingoCard` Class (15 Points)

Here is an example of a Bingo card (on a console screen):

```
  B  I  N  G  O
+--+--+--+--+--+
| 5|19|40|55|62|
+--+--+--+--+--+
|14|16|45|59|61|
+--+--+--+--+--+
| 6|27|XX|52|66|
+--+--+--+--+--+
|12|29|37|58|70|
+--+--+--+--+--+
| 4|28|42|53|63|
+--+--+--+--+--+
```

As you may notice above, a `BingoCard` looks like a two-dimensional array of size 5×5 of `BingoNumber` objects. According to the Bingo card discussed previously, the numbers in each row are randomly generated. Each row has its own range of numbers (see description above). A user can mark numbers on the card. With these requirements, here are implementation details:

- **Constructors**:

  - `public BingoCard()`: constructs a `BingoCard` object where all 24 numbers are objects of type `BingoNumber` constructed using randomly generated numbers. Do not forget that there is no `BingoNumber` in the middle of a Bingo card.

- **Accessor Methods**:

  - `public String toString()`: returns a string representation of a Bingo card. An example is the one shown above which is the output of the following code snippet:

```
BingoCard aCard = new BingoCard();
System.out.println(aCard);
```

- `public BingoNumber getNumber(int row, int column)`: returns the reference to the `BingoNumber` object at a given row and a given column.
- `public String toSideBySideString()`: returns the string representation of the Bingo card side-by-side, one with marks, and one without. This string allows the host to verify whether a player actually wins.

- **Mutator Methods**:
  - `public boolean mark(int number)`: marks a given `number` on the card and returns whether the given number is successfully marked. Note that a user can mark a number that is not on the card. If that is the case, this method will mark no number and simply returns `false`. The following is an output of this code snippet:

```
BingoCard aCard = new BingoCard();
System.out.println(aCard);
aCard.mark(45);
aCard.mark(22);
System.out.println(aCard);
System.out.println(aCard.toSideBySideString());
```

The output of the code snippet above is shown below. Note that the number 45 is on the card but not 22.

```
  B  I  N  G  O
+--+--+--+--+--+
| 5|19|40|55|62|
+--+--+--+--+--+
|14|16|45|59|61|
+--+--+--+--+--+
| 6|27|XX|52|66|
+--+--+--+--+--+
|12|29|37|58|70|
+--+--+--+--+--+
| 4|28|42|53|63|
+--+--+--+--+--+
  B  I  N  G  O
+--+--+--+--+--+
| 5|19|40|55|62|
+--+--+--+--+--+
|14|16|XX|59|61|
+--+--+--+--+--+
| 6|27|XX|52|66|
+--+--+--+--+--+
|12|29|37|58|70|
+--+--+--+--+--+
| 4|28|42|53|63|
```

```
+--+--+--+--+--+
  B  I  N  G  O     B  I  N  G  O
+--+--+--+--+--+  +--+--+--+--+--+
| 5|19|40|55|62|  | 5|19|40|55|62|
+--+--+--+--+--+  +--+--+--+--+--+
|14|16|XX|59|61|  |14|16|45|59|61|
+--+--+--+--+--+  +--+--+--+--+--+
| 6|27|XX|52|66|  | 6|27|XX|52|66|
+--+--+--+--+--+  +--+--+--+--+--+
|12|29|37|58|70|  |12|29|37|58|70|
+--+--+--+--+--+  +--+--+--+--+--+
| 4|28|42|53|63|  | 4|28|42|53|63|
+--+--+--+--+--+  +--+--+--+--+--+
```

For the side-by-side string, the card on the left is the one that is marked, and the one on the right shown the unmarked version of the card for the host to verify.

- **Instance Variables**: Again, think about this yourselves what are the set of instance variables and their types should be.

Part III: The `BingoBall` and `BingoCage` Classes (5 and 10 Points)

A Bingo ball is simply a ball with a number. A Bingo cage when constructed, it will contain 75 **unique** Bingo balls (ball numbers 1 through 75). A user can draw Bingo balls out of this cage one at a time. Each time a Bingo ball will be randomly picked. Note that once a ball is drawn, it cannot be redrawn again until the Bingo cage is reseted (put all Bingo balls back into the cage).

For the `BingoBall` class, it must have at least the following constructor and methods:

- `public BingoBall(int number)`: This is a **contstructor** that constructs a Bingo ball with a given number.
- `public int getNumber()`: This method returns the integer representation of a Bingo ball (e.g., the ball number).
- `public String toString()`: returns the string representation of the number of the Bingo ball.

For the `BingoCage` class, it must have the following methods:

- `public BingoCage()`: This is a **constructor** that constructs a Bingo cage containing 75 unique Bingo balls (75 objects of type `BingoBall`).
- `public BingoBall draw()`: This method randomly returns a Bingo ball (object of type `BingoBall`). Note that if there are no more Bingo balls left, this method should return `null`.
- `public void reset()` which resets the cage back to the original 75 unique Bingo balls.

Part IV: The `BingoPlayer` Class (30 Points)

This class represents a Bingo player which has the following abilities:

- A player has his/her first and last name
- A player will have his/her initial amount of money when he/she walks into a Bingo house.
- A player will have a hand that can hold any number of Bingo card.
- A player can be ordered to pay for a number of Bingo cards.
- A player can receive a number of Bingo cards to put into his/her hand.
- A player can mark numbers on his/her Bingo cards.
- A player can recognize whether a card is a winning card (BINGO!).
- A player can show his/her winning card(s) to the house to verify.
- A player can receive an amount of US Dollar if he/she wins a round.
- A player can discard all Bingo cards from his/her hand.

This class will be a little bit big. Here are implementation details:

- **Constructor**:
  - `public BingoPlayer(String first, String last, double amount)` constructs a Bingo player with with a given `first` name, `last` name, and the initial `amount` of money. Note that initially the hand of a Bingo player is empty (no Bingo cards)
- **Accessors**:
  - `public String getFullName()` returns the string representation of the full name of a Bingo player (first name a blank and last name)
  - `public String getFullInfo()` returns the string representation of the full information (full name and the current amount of money) of a Bingo player (see example below)
  - `public String toString()` returns a string representation of a Bingo player together with his/her Bingo cards in his/her hand.

    The following code snippet below uses the first three accessors above where the `setBingoCards()` method will be discussed later:

```
Person john = new Person("John", "Smith", 50.0);
BingoPlayer aPlayer = new BingoPlayer(john);
System.out.println("getFullName(): " + aPlayer.getFullName());
System.out.println("getFullInfo(): " + aPlayer.getFullInfo());
System.out.println("====");
System.out.println(aPlayer);
System.out.println("====");

BingoCard[] cards = new BingoCard[3];
cards[0] = new BingoCard();
```

```
cards[1] = new BingoCard();
cards[2] = new BingoCard();
aPlayer.setBingoCards(cards);
System.out.println(aPlayer);
```

The above code snippet should give you the following output (Bingo cards will be different):

```
getFullName(): John Smith
getFullInfo(): John Smith ($50.0)
====
John Smith ($50.0)
No Bingo Cards
====
John Smith ($50.0)
  B  I  N  G  O     B  I  N  G  O     B  I  N  G  O
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|13|25|37|56|63|  |10|29|31|52|71|  | 4|19|36|50|74|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 3|20|31|60|66|  | 3|21|38|58|70|  |11|18|34|55|64|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|11|22|XX|49|73|  | 8|23|XX|56|63|  | 6|25|XX|48|68|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 8|24|40|55|75|  | 6|17|36|53|72|  | 9|28|35|57|63|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|15|27|35|59|67|  | 2|20|44|59|68|  | 5|27|38|51|73|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
```

- `public boolean isBingo()` returns `true` or `false` whether the player has one or more winning card(s).
- `public BingoCard[] getBingoCards()` returns an **array** of type `BingoCard` that are winning cards. This method will be called by the host to verify whether the card(s) is the winning card(s).

- **Mutators**:
  - `public int remove(int amount)` removes a given `amount` in US dollars from a Bingo player. **Note** that if the player has enough fund, the return value will be the exact amount to remove. Otherwise, the return value will be the largest integer value that the player can play (the money that he/she has rounded down to integer). **Do not forget that this method must adjust the amount of money that the player has**.
  - `public void add(double amount)` adds more money into the player's pocket.
  - `public void addBingoCards(BingoCard[] cards)` adds Bingo cards into the player's hand in the form of an array of `BingoCard`. Note that the parameter is an **array** of type `BingoCard`. The size of the array must be exactly the number of Bingo cards that the player wants to buy (and have enough money).
  - `public void marks(int number)` marks all Bingo cards in the player's hand that contain the given `number`.

– `public void clear()` clears the player's hand which must be called before starting a new round of game.

Part V: The `Bingo` Class (35 Points)

This will be the `main` class. Note that this class must utilize objects of classes discussed earlier (OOP). There are a lot of places that the program will as for user inputs. For simplicity, we assume that the user will always enter a valid (type and range) inputs.

The `Bingo` program should perform the following:

1. Read the provided file named `players.txt`. Here is the content of the file:

```
50.00
10
James,Smith,50.0
Michael,Smith,50.0
Robert,Smith,50.0
Maria,Garcia,50.0
David,Smith,50.0
Maria,Rodriguez,50.0
Mary,Smith,50.0
Maria,Hernandez,50.0
Maria,Martinez,50.0
James,Johnson,50.0
```

The first line is a string representation of the amount of money the house currently has. The next line is a string representation of the number of players in the file. Each line (after the first line) is organized as `first,last,amount`. Your job is to read this file and construct all `BingoPlayer` objects from this file and put it into either an array or `ArrayList<BingoPlayer>`. Note that we will test your program with different player file. So, make sure you utilize the number of player listed in the file correctly. Next is to **Sort** the player in ascending order (first name then last name) and display all players on the console screen. Here is an example (amount may be different):

```
*****************************
* Welcome to SCI Bingo House *
*****************************
These are all available players:
1. David Smith ($46.0)
2. James Johnson ($50.0)
3. James Smith ($50.0)
4. Maria Garcia ($50.0)
5. Maria Hernandez ($46.0)
6. Maria Martinez ($50.0)
7. Maria Rodriguez ($50.0)
8. Mary Smith ($52.0)
9. Robert Smith ($50.0)
10. Michael Smith ($50.0)
```

2. Ask each Bingo player whether he/she wants to play in this round

   - If the player says no (`n`) move on to the next player
   - If the player says yes (`y`), ask how many Bingo cards (up to 4 cards) he/she wants to buy
     - If the player says between 1 and 4, take the money from the player. Do not forget to check the amount of money in case the player does not have enough money. Then give the player the appropriate number of Bingo cards.

   If no players wants to play in this round, just exit the program. Here is an example:

```
David Smith, would you like to play this round? (y/n): y
How many Bingo card would you like to buy? (1 - 4): 4
James Johnson, would you like to play this round? (y/n): n
James Smith, would you like to play this round? (y/n): n
Maria Garcia, would you like to play this round? (y/n): y
How many Bingo card would you like to buy? (1 - 4): 4
Maria Hernandez, would you like to play this round? (y/n): n
Maria Martinez, would you like to play this round? (y/n): n
Maria Rodriguez, would you like to play this round? (y/n): y
How many Bingo card would you like to buy? (1 - 4): 4
Mary Smith, would you like to play this round? (y/n): n
Robert Smith, would you like to play this round? (y/n): n
Michael Smith, would you like to play this round? (y/n): n
```

3. Make sure that the Bingo cage has all 75 balls (either reset the old object or construct a new one)

4. Show the amount of money of the house, the amount of money in the pot, the list of ball numbers (should be empty at the beginning) and show only those players that are playing (together with his/her hand) in this round

```
House: $62.0
Pot: $6.0
Balls: []
1. David Smith ($42.0)
  B  I  N  G  O     B  I  N  G  O     B  I  N  G  O     B  I  N  G  O
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 6|27|45|57|68|  | 6|19|45|47|65|  |13|23|40|53|75|  | 8|22|43|57|65|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 4|28|37|46|62|  |10|21|38|50|67|  | 8|24|35|56|71|  | 6|19|33|51|69|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 7|18|XX|55|70|  | 3|24|XX|57|61|  | 3|17|XX|55|67|  | 2|28|XX|48|73|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|12|19|38|47|72|  | 8|16|36|58|63|  | 1|20|38|48|61|  |13|27|41|53|68|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 5|16|36|50|73|  |11|25|35|49|72|  | 5|26|42|54|63|  | 1|24|40|56|67|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
2. Maria Garcia ($46.0)
```

```
  B  I  N  G  O     B  I  N  G  O     B  I  N  G  O     B  I  N  G  O
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|11|23|37|55|68|  |10|30|35|60|66|  |14|28|33|55|64|  | 1|22|41|52|62|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 7|22|32|60|66|  | 7|19|37|58|75|  |10|23|45|51|68|  | 3|25|37|49|74|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 1|26|XX|51|69|  | 8|26|XX|51|74|  | 4|22|XX|53|72|  | 8|24|XX|54|66|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|10|16|40|53|63|  | 9|16|38|53|68|  | 5|27|41|60|65|  |12|29|35|59|68|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|15|18|33|57|71|  | 6|27|42|57|64|  |12|16|43|47|70|  | 6|20|40|51|73|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
3. Maria Rodriguez ($46.0)
  B  I  N  G  O     B  I  N  G  O     B  I  N  G  O     B  I  N  G  O
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 6|20|45|57|65|  | 7|28|42|60|65|  | 1|26|44|51|66|  |15|16|36|46|62|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 5|28|31|49|66|  | 2|26|36|56|63|  |13|24|40|57|74|  | 1|17|34|55|68|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 1|17|XX|60|73|  | 4|23|XX|51|62|  |15|17|XX|59|67|  |13|30|XX|60|70|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
|15|16|34|53|75|  |11|29|44|54|69|  |12|23|43|52|65|  |12|27|39|52|72|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
| 4|27|38|50|69|  | 5|18|40|58|72|  |14|25|41|56|61|  |11|20|31|51|69|
+--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+  +--+--+--+--+--+
There is no winning card yet.
```

5. There are two options from here; (1) draw one ball and check the output or (2) draw balls until a player gets a BINGO. So, ask the user:

```
1) Draw a ball, 2) Draw balls until Bingo:
```

6. Draw a Bingo ball from the Bingo cage.

7. Tell each player in this round to mark the number (ball's number)

8. Check each player in this round whether he/she have a winning card

   - If a player has a winning card, go to step 9.
   - If no player has a winning card, go back to step 4 if the user chooses choice number 1 (Draw a ball). Go to step 4 (but skip step 5) if the user chooses choice number 2 (Draw balls until Bingo).

9. Show the winning card (side-by-side) on the console screen for the house to verify.

```
Maria Rodriguez says this card is a winning card
  B  I  N  G  O     B  I  N  G  O
+--+--+--+--+--+  +--+--+--+--+--+
|XX|XX|36|46|XX|  |15|16|36|46|62|
```

```
+--+--+--+--+--+  +--+--+--+--+--+
|XX|XX|34|55|68|  | 1|17|34|55|68|
+--+--+--+--+--+  +--+--+--+--+--+
|XX|30|XX|60|XX|  |13|30|XX|60|70|
+--+--+--+--+--+  +--+--+--+--+--+
|XX|27|39|52|72|  |12|27|39|52|72|
+--+--+--+--+--+  +--+--+--+--+--+
|XX|20|31|XX|XX|  |11|20|31|51|69|
+--+--+--+--+--+  +--+--+--+--+--+
These are balls that have been drawn so far in this round:
[25, 38, 71, 16, 22, 62, 18, 61, 51, 3, 49, 12, 13, 21, 45, 40, 1, 69, 7,
35, 54, 15, 50, 4, 70, 6, 19, 17, 11]
Is it a winning card? (y/n):
```

Note that answer y or n is not important here. It is just for us to double check your work.

10. Give the amount of money in the pot to the winner
11. Update the file `players.txt`. For simplicity, just simply create the new file named `players.txt` to replace the old one but use exact same format.
12. Go back to stop 1.

An example of an output of a run can be found in the file `output2.txt`.

Submission

The due date of this project is stated on the CourseWeb. Late submissions will not be accepted. For this project, you have to submit the following files:

- `BingoNumber.java`
- `BingoCard.java`
- `BingoBall.java`
- `BingoCage.java`
- `BingoPlayer.java`
- `Bingo.java`

Zip them in to one file named `project3.zip` and submit the file `project3.zip` via CourseWeb.