# COMP151 Lab11

Your task is to implement solutions to the following two independent projects described below.

# Implement ADT set

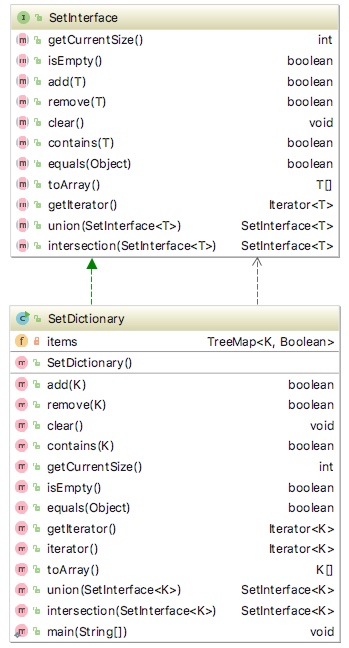
### Problem Description

We want to implement the ADT set. Recall from Chapter 1 that a set is an unordered collection of objects where duplicates are not allowed. The operations that a set should support are:

* Add a given object to the set (will always return true)
* Remove a given object from the set
* See whether the set contains a given object
* Clear all objects from the set
* Get the number of objects in the set
* Return an iterator to the set
* Return a set that combines the items in two sets (the union)
* Return a set of those items that occur in both of two sets (the intersection). The algorithm **must** take advantage of the fact that **both lists are sorted**

These operations are defined in the provided SetInterface. Implement the class SetDictionary.java that uses a dictionary internally (instance of class TreeMap) to implement the SetInterface (the skeleton of the SetDictionary.java class is provided, and it contains main that you can use to test your methods). The ***key*** is the item, the ***value*** is a boolean set to true.

See the UML diagram and the sample runs below



### Sample run of the program:

# CREATING set1

# --> set1 has 5 items:

# [-1, 0, 1, 2, 3]

# --> contains for -1 yields true

# --> contains for -2 yields false

# --> contains for 3 yields true

# --> contains for 4 yields false

# --> Added 1 again to the set1, should be the same

# --> set1 has 5 items:

# [-1, 0, 1, 2, 3]

# --> Iterating over set1 utilizing getIterator method

# -1

# 0

# 1

# 2

# 3

# --> Iterating over set1 utilizing iterator method

# -1

# 0

# 1

# 2

# 3

# --> Iterating over set1 utilizing forEach lambda

# -1

# 0

# 1

# 2

# 3

# --> Removing -1 20 3 from set1:

# ---> -1 was removed - CORRECT

# ---> 20 was not removed - CORRECT

# ---> 3 was removed - CORRECT

# --> Should just have 0 1 and 2 now

# --> set1 has 3 items:

# [0, 1, 2]

# CREATING set2

# --> set2 has 6 items:

# [-1, 1, 2, 3, 5, 8]

# set1 and set2 are NOT the same - CORRECT

# CREATING UNION OF set1 and set2

# --> The union of set1 and set2 has 7 items: [-1, 0, 1, 2, 3, 5, 8]

# --> set1 should be unchanged

# --> set1 has 3 items:

# [0, 1, 2]

# --> set2 should be unchanged

# --> set2 has 6 items:

# [-1, 1, 2, 3, 5, 8]

# CREATING UNION OF set1 and set1

# --> The union of set1 and set1 has 3 items: [0, 1, 2]

# set1 and testUnion are the same - CORRECT

# CREATING INTERSECTION OF set1 and set2

# Finished creating intersection.

# --> The intersection of set1 and set2 has 2 items: [1, 2]

# --> set1 should be unchanged

# --> set1 has 3 items:

# [0, 1, 2]

# --> set2 should be unchanged

# --> set2 has 6 items:

# [-1, 1, 2, 3, 5, 8]

# CREATING INTERSECTION OF set2 and set1

# Finished creating intersection.

# --> The intersection of set2 and set1 has 2 items: [1, 2]

# --> set1 should be unchanged

# --> set1 has 3 items:

# [0, 1, 2]

# --> set2 should be unchanged

# --> set2 has 6 items:

# [-1, 1, 2, 3, 5, 8]

# CREATING INTERSECTION OF set2 and set2

# Finished creating intersection.

# --> The intersection of set2 and set2 has 6 items: [-1, 1, 2, 3, 5, 8]

# --> set1 should be unchanged

# --> set1 has 3 items:

# [0, 1, 2]

# --> set2 should be unchanged

# --> set2 has 6 items:

# [-1, 1, 2, 3, 5, 8]

# set2 and testIntersection are the same - CORRECT

# CREATING INTERSECTION OF testUnion and set2

# Finished creating intersection.

# --> The intersection of testUnion and set2 has 2 items: [1, 2]

# --> testUnion should be unchanged

# --> testUnion has 3 items:

# [0, 1, 2]

# --> set2 should be unchanged

# --> set2 has 6 items:

# [-1, 1, 2, 3, 5, 8]

# set2 and testIntersection are NOT the same - CORRECT

# CREATING set3

# --> set3 has 7 items:

# [-1, 0, 2, 5, 7, 9, 11]

# CREATING set4

# --> set4 has 6 items:

# [1, 2, 3, 4, 5, 9]

# CREATING INTERSECTION OF set3 and set4

# Finished creating intersection.

# --> The intersection of set3 and set4 has 3 items: [2, 5, 9]

# Process finished with exit code 0

# Implement Bingo Game

### Problem Description

In the game of Bingo, numbers are chosen at random from a set between 1 and 75 inclusive. The numbers in the range:

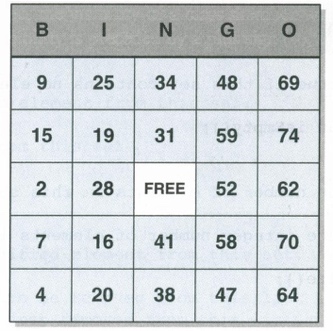
* 1 to 15 are associated with the letter B
* 16 to 30 are associated with the letter I
* 31 to 45 are associated with the letter N
* 46 to 60 are associated with the letter G
* 61 to 75 are associated with the letter O

The person managing the game (“the caller”) selects a number from the Bingo drum randomly, and then announces the letter and the number. The caller then sets aside that number so that it cannot be used again in that game. All the players then mark any squares on their card that match the letter and the number called. Once any player has five squares in a row, they announce “BINGO!” and claim their prize.

For this program however, please note that:

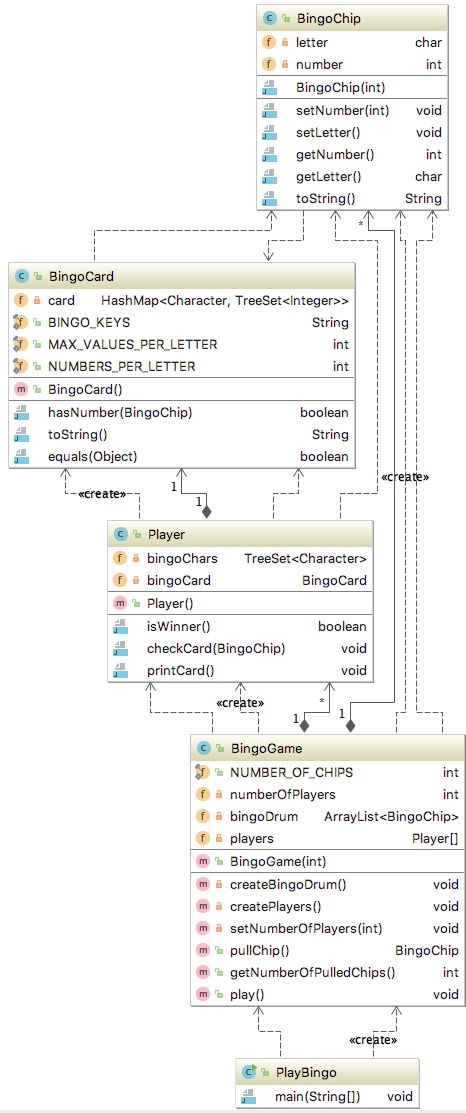
1. The player can announce “BINGO” as soon as (s)he gets one number for each letter, not necessarily in a row. For example, 2 25 31 58 70 would be a winning combination.
2. Multiple winners are possible.

The following picture shows a sample bingo card:



**33**

Please have a look at the attached UML diagram, it contains five classes. Below are their highlights:



1. BingoChip class
   1. constructor
      1. sets the number to the given value by calling mutator method
      2. calls setLetter method which based on the given number, sets the letter to the appropriate value (B, I, N, G, or O); **must utilize a formula** to calculate the letter
2. BingoCard class
   1. constructor - creates a card for a player – the structure is a dictionary were the *key* is a letter and the *value* is the collection of five unique numbers from the appropriate range
   2. hasNumber method – checks if the called number is on the card, if it is, adds a marker (integer of 0) to the values for the given letter
   3. toString method – creates a String representation of the card, utilizes String.format method to format the content properly (please look at the sample run and note the order)
   4. equals method – returns true if this card is the same as the other
3. Player class
   1. constructor
      1. creates bingoCard
      2. prints the created card
      3. creates bingoChars object that the player will use to collect the B, I , N, G, O letters
   2. checkCard method – checks if the called number is on the card, if it is, adds the letter to bingoChars
   3. isWinner method – checks if the player had at least one number for each letter (size of the bingoChars is the same as the length of BINGO\_KEYS
   4. printCard method – utilizes bingoCard toString() method
4. BingoGame class
   1. constructor:
      1. sets the number of players for the game
      2. creates bingoDrum – ArrayList that contains BingoChip objects
      3. creates players - array of Player objects
   2. pullChip method – pulls randomly one chip from the bingoDrum
   3. getNumberOfPulledChips method – returns the number of chips pulled
   4. play method – plays the game until BINGO is announced:
      1. pulls a chip and calls the chip letter and the number
      2. for each player checks if the player’s card has the called number
      3. prints each player’s card
      4. checks each player’s card for the winning combination and announces the BINGO if appropriate
      5. once the game is over it prints the number of chips that were called
5. PlayBingo class has the main method
   1. gets user input with the number of players (if negative number is entered 1 is assumed)
   2. creates the game object
   3. starts the game

### Sample run of the program:

---> Setting up bingo game.

Enter number of players.

4

---> Creating bingo card for Player 1

B 1 4 5 10 14

I 17 20 25 26 27

N 32 35 39 43 45

G 46 48 51 55 56

O 63 66 68 69 71

---> Creating bingo card for Player 2

B 5 6 9 10 15

I 17 20 25 29 30

N 31 32 38 39 43

G 46 47 49 50 57

O 63 70 71 72 74

---> Creating bingo card for Player 3

B 4 6 10 11 15

I 18 19 21 23 29

N 35 36 40 41 42

G 48 50 55 56 57

O 63 67 69 73 74

---> Creating bingo card for Player 4

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Starting the game with 4 players:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

---> Calling: O 63

Player's 1 card:

B 1 4 5 10 14

I 17 20 25 26 27

N 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 5 6 9 10 15

I 17 20 25 29 30

N 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 18 19 21 23 29

N 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: G 54

Player's 1 card:

B 1 4 5 10 14

I 17 20 25 26 27

N 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 5 6 9 10 15

I 17 20 25 29 30

N 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 18 19 21 23 29

N 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: N 39

Player's 1 card:

B 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 5 6 9 10 15

I 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 18 19 21 23 29

N 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: O 71

Player's 1 card:

B 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 5 6 9 10 15

I 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 18 19 21 23 29

N 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: I 19

Player's 1 card:

B 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 5 6 9 10 15

I 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 0 18 19 21 23 29

N 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: N 35

Player's 1 card:

B 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 5 6 9 10 15

I 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 0 18 19 21 23 29

N 0 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: B 5

Player's 1 card:

B 0 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 0 5 6 9 10 15

I 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 0 18 19 21 23 29

N 0 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: N 33

Player's 1 card:

B 0 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 0 5 6 9 10 15

I 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 0 18 19 21 23 29

N 0 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 0 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: I 30

Player's 1 card:

B 0 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 0 5 6 9 10 15

I 0 17 20 25 29 30

N 0 31 32 38 39 43

G 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 0 18 19 21 23 29

N 0 35 36 40 41 42

G 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 0 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

---> Calling: G 50

Player's 1 card:

B 0 1 4 5 10 14

I 17 20 25 26 27

N 0 32 35 39 43 45

G 46 48 51 55 56

O 0 63 66 68 69 71

Player's 2 card:

B 0 5 6 9 10 15

I 0 17 20 25 29 30

N 0 31 32 38 39 43

G 0 46 47 49 50 57

O 0 63 70 71 72 74

Player's 3 card:

B 4 6 10 11 15

I 0 18 19 21 23 29

N 0 35 36 40 41 42

G 0 48 50 55 56 57

O 0 63 67 69 73 74

Player's 4 card:

B 6 11 12 13 15

I 24 25 26 27 28

N 0 32 33 42 43 45

G 47 52 53 56 59

O 61 62 67 70 74

!!! Player 2 says BINGO !!!

10 chips were called.