CMPT225, Spring 2023
Homework Assignment 1
Due date: Friday, January 27, 2023, 23:59

You need to implement the following classes:
Part 1:
- *myiterators.ArrayIterator.java*
- *myiterators.PrimeNumbersIterator.java*
- *myiterators.RangeIterator.java*

Part 2:
- *fifteenpuzzle.FifteenPuzzle.java*
- *fifteenpuzzle.IllegalMoveException.java*
- *fifteenpuzzle.BadBoardException.java*

Note that the files for Part 1 must be in under the package (folder) **integeriterators/**,
and the files for Part 2 must be under the package (folder) **fifteenpuzzle/**.
You may add more classes to your solution if necessary.

Submit all your files in ***assignment1.zip*** to CourSys. Make sure your zip file can be unzipped using the command "*unzip assignment1.zip*" in CSIL. All your solutions in the zip file must be under the **src** folder, same as in the provided *hw1-cmpt225-spring23.zip*.

Compilation: Your code MUST compile in CSIL using javac.
Make sure that your code compiles without warnings/errors.
If the code does not compile in CSIL the grade on that part will be 0 (zero).
Even if you can't solve a problem completely, make sure it compiles.

The assignment will be graded mostly **automatically**, with some exceptions.

Discussion with others: You may discuss the assignment with your classmates/tutors (or anyone else), but coding must be entirely your own.

References: You may use textbooks, wiki, stack overflow, geeksforgeeks, etc. If you do, please specify the references in comments. Asking others for solutions (online or in person) is prohibited.

Readability: Your code should be readable using the standard Java conventions. Add comments wherever is necessary. If needed, write helper functions or add classes to improve readability.

Do not add main() to your solutions. The main() method will be in the test files.
Examples of such tests are provided in *TestIterators.java* and *TestFifteenPuzzle.java*.

Warnings: Warnings during compilation will reduce points. More importantly, they indicate that something is probably wrong with the code.

Testing: Test your code. Examples of tests are included. Your code will be tested using the provided tests as well as additional tests. You should create more tests to check your solution.

Good luck!

Part 1 [50 points] - IntegerIterator extends Iterator<Integer>

IntegerIterator represents a sequence (finite or infinite) of integers.
The interface has the following public methods:

`public boolean hasNext()` : Returns true if the sequence has more elements, and returns false otherwise. If the sequence is infinite, has Next() always returns true.
It is inherited from Iterator<Integer>

`public Integer next():` Returns the next element in the sequence.
It is inherited from Iterator<Integer>

`public void reset():` Resets the iterator to the first element of the sequence.

The interface is provided with the assignment under the package integeriterators.

Your goal is to write the following 3 classes implementing this interface:

1) **ArrayIterator [15 points]**
   The class has two constructors.
   1) `public ArrayIterator(int[] ar)`

   The constructor gets an array of ints. For this constructor the iterator goes through the elements of the array once in the given order.
   2) `public ArrayIterator(int[] ar, boolean isCircular)`
   - The constructor gets an array of ints and a boolean variable.
   - If `isCircular` is `false`, the iterator iterates through the elements of the array once in the given order.
   - If `isCircular` is `true`, the iterator iterates through the elements of the array in the given order in a circular manner. That is, when the iterator reaches the last element of the ar, it returns to the ar[0]. In this case hasNext() always returns `true`.

2) **RangeIterator [15 points]**
   The class has three constructors:
   1) `public RangeIterator()`

   Defines the *infinite* sequence of non-negative integers 0,1,2,3...
   2) `public RangeIterator(int s)`

   Defines the *infinite* sequence of non-negative integers s,s+1,s+2,s+3...
   3) `public RangeIterator(int s, int t)`

   Defines the *finite* sequence of non-negative integers s,s+1,s+2,s+3...t-1

3) PrimeNumbersIterator [20 points]

The PrimeNumbersIterator returns the next prime number.

The class implements the infinite sequence of prime numbers. It has two constructors:

```
1) public PrimeNumbersIterator()
```

Defines the sequence of all prime numbers in the increasing order: 2,3,5,7,11,13,17,19…

```
2) public PrimeNumbersIterator(int n)
```

Defines the sequence of prime numbers in the increasing order starting from the smallest prime ≥n. For example

- If n==7, the sequence is 7, 11, 13, 17, 19, 23, 29, 31…
- If n==8, the sequence is 11, 13, 17, 19, 23, 29, 31, 37…

Part 2 [50 points] - Fifteen Puzzle

FifteenPuzzle is a class representing the game Fifteen Puzzle. Find the game online and play it. See wiki page for the description https://en.wikipedia.org/wiki/15_puzzle

In this part you will read the initial board from a file, and move the tiles on the board.

The file is expected to have 4 rows each row representing one row of the board
- Each row has exactly 11 chars + '\n' for newline
- The 11 chars in the format "AA BB CC DD"
- AA represents the first number: it is either
  * one of the numbers 10…15 or
  * space followed by a number 1…9 (space has ascii value 32)
- Same format for the numbers BB, CC, DD
- There is one space (ascii value 32) between two numbers
- Each line ends with '\n' for newline (ascii code 10)

For an example, see board1.txt or board2.txt.

Write the class FifteenPuzzle with the following constructor and public methods:

```
public FifteenPuzzle(String fileName) throws IOException,
BadBoardException
```

The constructor gets a name of a file as an argument. It reads the file and initializes the board. It throws an exception if the file does not exist or if the board is not in the correct format. The class **BadBoardException** is provided with the assignment.

1)
```
public void makeMove(int tile, int direction)
            throws IllegalMoveException
```

The method moves the tile in the specified direction.
The possible directions are given by the constants provided in the class **FifteenPuzzle**:

```
FifteenPuzzle.UP
FifteenPuzzle.DOWN
FifteenPuzzle.LEFT
FifteenPuzzle.RIGHT,
```

If the move is illegal the method throws **IllegalMoveException**. The class **IllegalMoveException** is provided with the assignment.

2)
```
public boolean isSolved()
```

Returns true if the board is in the solved state.

3)
```
public String toString()
```

Returns the string in the same format as the file. For an example see the method testReadFromFile1() in TestFifteenPuzzle.java.