

CMPT225, Fall 2026
Homework Assignment 1
Due date: Friday, October 2, 2026, 23:59

You need to implement the following classes:

Part 1:

- *integeriterators/ArrayIterator.java*
- *integeriterators/RangeIterator.java*
- *integeriterators/PrimeNumbersIterator.java*

Part 2:

- *rushhour/RushHour.java*
- *rushhour/IllegalMoveException.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) **rushhour/**. 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 *cmpt225-hw1-fall26.zip*.

1. See the provided example for the expected format.
2. Use the provided *check_format.sh* to check that your zip has the expected folder structure.

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

References: You may use textbooks, wiki, stack overflow, geeksforgeeks, etc. If you do, specify the references in comments. Posting questions online asking for solutions is prohibited. Using AI tools for solving the assignment 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.

Compilation: Your code MUST compile in CSIL using javac. Make sure that your code compiles without warnings/errors. Even if you can't solve a problem completely, make sure it compiles.

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

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] - Integer Iterator

IntegerIterator represents a sequence (finite or infinite) of integers $a_0, a_1, a_2, \dots$.

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, hasNext() always returns true.

public int getNext() : Returns the next element in the sequence.

public void reset() : Resets the iterator to the initial element of the sequence, i.e., to a_0 .

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 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... (0 and 1 are not prime numbers). The following results are expected:

```
next(); -- returns 2
next(); -- returns 3
next(); -- returns 5
hasNext(); -- returns true
next(); -- returns 7
next(); -- returns 11
hasNext(); -- returns true
```

2) `public PrimeNumbersIterator(int n)`

Defines a sequence of prime numbers in the increasing order starting from the smallest prime $\geq n$. For example, when creating the iterator using `PrimeNumbersIterator(12)`, the following results are expected:

```
next(); -- returns 13, because 13 is the smallest prime  $\geq 12$ 
next(); -- returns 17
next(); -- returns 19
next(); -- returns 23
hasNext(); -- returns true
reset(); -- resets the iterator to 13
next(); -- returns 13
next(); -- returns 17
```

If we create the iterator using `PrimeNumbersIterator(7)`, the following results are expected:

```
hasNext(); -- returns true
next(); -- returns 7, because 7 is prime
next(); -- returns 11
next(); -- returns 13
next(); -- returns 17
reset(); -- resets the iterator to 7
next(); -- returns 7
```

[50 points] - Rush Hour

RushHour is a class representing the game RushHour.

See wiki page for the description [https://en.wikipedia.org/wiki/Rush_Hour_\(puzzle\)](https://en.wikipedia.org/wiki/Rush_Hour_(puzzle))

Find the game online and play it.

Your goal in this part is to read an initial board from a file, and move the cars on the board.

The file is expected to have 6 rows with 6 chars in each row.

- A dot represents an empty cell.
- Each car is represented by a letter.
- Each car can go either only horizontally or only vertically. The allowed direction of a car is defined by the longer side of the car.
- Vehicles can only be moved along a straight line on the grid; rotation is forbidden
- The red car (the one we need to get to the exit) is denoted by X.
- The goal is to bring the red car, denoted by X to the right edge of the board.
- You may assume that the file always represents a legal board.

See board1.txt for an example.

Write the class RushHour with the following constructor and three public methods:

1) `public RushHour(String fileName) throws FileNotFoundException`

The constructor gets a name of a file as an argument. It reads the file and initializes the board.

2) `public void makeMove(char carName, int direction, int length)
throws IllegalMoveException`

The method moves the car in the car for length steps in the given direction.

If the move is illegal the method throws IllegalMoveException. The move can be illegal because the direction is incorrect or because some cells on the way are occupied.

The class IllegalMoveException is provided with the assignment.

It is very simple, it just inherits from Exception.

3) `public boolean isSolved()`

Returns true if and only if the car X touches the right edge of the board.

Remarks:

The provided class RushHour has several constants: UP, DOWN, LEFT, RIGHT, and SIZE. They are provided for your convenience. You may use them , but you don't have to.

You may add additional private methods if needed. You may also add new classes if needed. For example, you may want to consider creating a class Car to represent a car on the board.