Focus

Instructions: Unless it has been done for you, print a copy of this document for your pair. As you work through it, write answers to any questions or prompts that are in **boldface**.

Your names:

Learning Objectives

Content Objectives

Parentheses below correspond to part of the knowledge units in the ACM's *Computer Science Curricula 2013*.

After completing this activity, students should be able to:

- Implement a double-ended queue using a linked data structure (SDF/Fundamental data structures).
- Implement the Iterable interface (SE/Design patterns).
- Use generic types (PL/Basic type systems).

Process Objectives

After completing this activity, students should have improved their ability to:

- Program in a pair. [Management]

Legal Notice

Model 1: Playing the Game

Open the Focus project in Eclipse. Run Focus.jar to play the game. It's up to you whether you want to read the rules or play the game first, but do both.

Focus was invented by the American game designer and collector Sid Sackson in 1963. It won the prestigious Spiel des Jahres (Game of the Year) award in 1981. Players move pieces (and piles of pieces) on a board, attempting to deny the opponent any legal move.

Rules

Players: 2

Equipment: The board is an 8x8 checkerboard with the three squares nearest each corner removed. Each player has 18 pieces in their color.

Object: Be the last player with a legal move.

Setup: See the running program.

Play: Black plays first, then play alternates.

On a turn, a player may either move some or all of one pile or play a piece from reserves.

To move, a player picks up a pile of pieces in one square. The top piece must belong to that player. If there are several pieces in the square, it is legal to pick up the top part of the pile, leaving one or more pieces behind. The picked up pieces are then moved horizontally or vertically and placed on top of another pile (or empty square). The distance moved must exactly equal the number of pieces moved. If, as a result, the destination pile is more than five pieces tall, pieces in excess of five are removed from the bottom of that pile. Pieces belonging to the moving player are added to her reserves; pieces belonging to the other player are simply removed from the game.

Instead of moving, a player may take a piece from her reserves and place it on any square on the board, on top of any pieces already there. Again, pieces in excess of five are either added to reserves or removed from the game.

Note that the four initially-empty squares along each side are legal places to play, but the three "missing" squares in each corner are not.

A player loses the game when she is unable to play on her turn because she has no reserves and all piles on the board belong to the opponent.

1. **What does the program do if you try to make an illegal move?**

2. **Does the program detect the end of the game?**

3. **Does the program detect the winner?**

4. **What strategy advice would you give?**

Model 2: Use of Deque

You are going to write the classes Deque, ListNode, and DequeIterator. Only Deque is used directly by the other classes; the other two are used by Deque.

A deque (pronounced like "deck") is a double-ended queue: it's like a queue, but allows insertion into and deletion from either end. Before building it, you will examine how it is used.

Examine the FocusModel class.

5. **What is the type of the field `board`?**

6. **What is the type of the elements of each Deque?**

7. **The constructor defines a String `setup`. What is the purpose of this String?**

8. **In `setup`, what does a `b` signify?**

9. **What does a `w` signify?**

10. **What does a . signify?**

11. **What does a ? signify?**

12. **How many Deques does the main method create?**

13. **The for loop on lines 159-161 picks up some pieces from the source Deque and puts them into the "hand", also represented as a Deque. From which end of the source are pieces removed?**

14. **To which end of the hand are pieces added?**

15. **When the pieces are added to the destination Deque (lines 164-166), to which end are they added?**

16. **When excess pieces are removed from the destination Deque (lines 168-173), from which end are they removed?**

17. **Does the "top" of a pile (as it is graphically displayed) correspond to the front or back of the Deque that represents it?**

Model 3: Deque

Implement the Deque. To do this, go through the tests in DequeTest.java in the order in which they appear in that file.

Your implementation must be linked, not array-based. You may wish to start by examining a linked implementation of a regular FIFO queue.

You will need two auxiliary classes, perhaps named Node and DequeIterator. Your class will also have to implement the Iterable interface.

Be sure to play the game (by running Focus.java) a few times after you're done to make sure the entire system is working correctly.