GAME OVERVIEW:

I made a game called Bagh Chal. It is a classic game in Nepal, and it works similarly to checkers.

The board is made of a 5x5 grid connected by lines such that some moves are valid and some are not.

The tigers try to eat 5 goats by jumping over them, and the goats try to trap the tigers.

In my version, the player is in charge of the goats.

The game begins with tigers in the 4 corners, and 20 goats in the player’s hand.

Players take turns placing one goat, then moving a tiger until all the goats have been placed.

Then, the player selects a goat on the board and moves it.

The game continues until one of the players achieves his or her goal.

4 CONCEPTS

2D ARRAYS

I use 2D arrays to form the map of tiles. The board is a 5x5 grid.

My 2D Array was formatted such that arr[col][row] goes to column col and row row.

I changed my implementation somewhat from my proposal,

in that I created an EmptyTile subclass of Tile.

In my game, parts of the board that are not filled with an animal are filled with this EmptyTile.

That way, no slots are null.

COLLECTIONS

I used a LinkedList in order to implement an Undo() function.

The LinkedList was the best collection to serve this purpose because I could keep track of the

order of moves and pop them off the list accordingly.

I created a Board class with all the properties of the Board after a certain move.

The list held instances of this Board class.

To undo, I popped a Board off the list and updated the GameTable properties accordingly.

INHERITANCE/SUBTYPING FOR DYNAMIC DISPATCH

I used inheritance/subtyping for dynamic dispatch in my Tile classes.

I created one abstract class Tile, with three subclasses: EmptyTile, GoatTile, and TigerTile.

I used an abstract class instead of an interface, since some parts of isValidMove(), draw(), and

changeTiles() were used the same code, and the tiles had the consistent fields besides color

The inheritance was needed and not easily represented with an enum/variable. I changed the method

isValidMove() for all 3 subclasses such that Empty is always false, and goat and tiger use super.

isValidMove(), move(), isEmpty(), isGoat(), and isTiger(), and draw() all used dynamic dispatch

The methods isEmpty(), isGoat(), and isTiger() were simple, but allowed for easy identification.

I overrode the draw method in GoatTile to indicate when the tile was selected.

I added extra methods to the tiger class in isJump() and jump().

I added extra methods to the goat class in getSelected() and flipSelected().

I added the extra field selected to the goat class.

TESTABLE COMPONENT

The main state of my game is the GameTable. I tested smaller methods like move() and jump()

extensively, but I also tested parts of the main state including methods that respond to user

input, the results of the undo button, and the effects of each player winning the game.

I tested edge cases including pressing undo after no moves, after one move, after multiple moves,

after selecting one, after selecting multiple. I tested all possible jumps, including invalid ones,

players attempting to move into invalid squares, players selecting multiple goats before moving,

players attempting to select non-goat tiles.

All actions occurred through the GameTable object, so testing the methods in this tangentially tests

the other functions and methods as well.

CLASS DESCRIPTIONS

Board

The Board class holds the state of the game board after a certain move. The purpose of this class

is to allow for the undo button.

EmptyTile

The empty tiles hold no animals. They are able to be moved into, but cannot be moved or selected

themselves

Game

Holds the main method and directions for creating the GUI. This class includes the instructions

pop up, and the layout visible to the user.

GameTable

This class holds most of the functionality of the program. The actionListeners are added in this

class. It holds a map of Tiles, functions to affect this map, and the ability to draw what the

user ends up seeing

GoatTile

The goat tile holds a goat animal. These are placed by the user, and after 20 have been placed

they can be selected and moved into empty tiles.

TigerTile

The tiger tile holds a tiger animal. These move independently by the CPU. Unlike the other tiles,

tiger tiles can jump to "eat" goats.

Tile

This is the abstract class containing common fields and methods for all Tiles.