Implementing Kalah with a Changeable Object-Oriented Design

Matthew Eden
Department of Electrical, Computer and Software Engineering
University of Auckland
Auckland, New Zealand
mede607@aucklanduni.ac.nz

***Abstract*—There were several decisions made in designing Kalah (a.k.a Mancala) in Java 7 such that it supported changeability. The main aspects of the design considered**

***Index Terms*—kalah, mancala, object-oriented design, changeability, java**

I. Introduction

This document outlines the design decisions made when designing an implementation of the game Kalah (known elsewhere as Mancala) in Java 7. There were three aspects of design considered during development, and these aspects were considered with great concern as to how they affect the changeability of the object-oriented design. The first aspect of the design concerned the number of 'houses' available to a player. The prescribed set of rules dictated *six* of these 'houses' for each player, however it was decided that this number should be changeable to accomodate future rulesets. The second aspect of the design concerned the number of 'seeds' that each house starts the game with. The prescribed set of rules dicated that this starting value should be *4* seeds per house, however the design should allow for ruleset changes. The third aspect of the design concerned the separation of input/output and game logic. This is to ensure future alterations to the specification of the output that was originally provided can be executed without needing to consider implications on the functionality of the game itself.

II. Design Decisions

A. Number of Houses

The design I have implemented allows any number of houses (a minimum of 1 house) to be used. This supports changeability

B. Number of Seeds

My design allows a developer to easily alter the number of seeds in each house at the start of the game. In addition, there is also support for altering the amount of seeds that are dropped into each pit when a player makes a move.

C. Separation of IO and Game Logic

My design makes use of a class *GameIO*, which is concerned with handling player input and console output. This supports changeability as it means future alterations to IO are constrained to a single class.