CSC2002S — Nkosingiphile Gumede

Assignment 2 - Concurrency 1 — GMDNKO003

1.5) Report

1.5.1) Description of classes

- **DriverRangeApp:**

  The DriverRangeApp in the main class of the program. It prescribes the number of golfer threads that will be set running and initializes the shared variables (sharedStash and sharedField). It also prescribes the size of the golfers' buckets. It is a single thread which runs, sleeps, and then makes the other thread's (of the Golfer and Bollie classes) runs come to completion at its completion.

- **Golfer:**

  The Golfer class represents the number of golfer threads that will be run in the program. Each one of these golfer threads will collect the number of balls necessary to occupy the full capacity of their respective buckets from the stash. They will then hit each one of these balls at random intervals until their respective buckets are empty. At this point the golfers will have to wait until the stash has been refilled with the balls collected by Bollie. When Bollie enters the golf range to collect the balls, the golfer threads have to wait until such a point that Bollie is off the field.

- **Bollie:**

  Bollie is the automated ball collection system of the golf range represented by a single thread which sleeps and then continues at randomly spaced time intervals. When Bollie takes to the field, it notifies the golfer threads, in order for then to wait. When Bollie has completed the collection of balls and added them to the stash, it notifies the golfer thread in order for them to continue.

- **BallStash:**

  The ball stash is the stash area in the golf range where golf balls are stored. It is representing by a list of objects of class golfBalls. It is initialized to the prescribed size of the stash. This is where golfers come to replenish their buckets when empty. This is also where Bollie deposits the golf balls after collecting them.

- **Range:**

  The range is the field on which the golf balls land after being hit. It is represented by a list of objects of class golfBalls. It is initialized to 0 and added to when a golfer hits their ball onto the field. This is where Bollie comes to collect the balls that have been hit by the golfers. When Bollie enters the range, Bollie is required to collect all of the balls that are on the field.

- **golfBalls:**

  The golfBalls class represents golf ball objects. Each golf ball is required to have a unique ID in order to enable the tracking golf balls hit onto and collected from the range.

1.5.2) Concurrency Requirements

Due to multithreading, this program is likely to face a number of concurrency issues. Mutual exclusion and deadlock prevention are required to ensure thread safety. The usage of atomic variables, method synchronization and barriers ensure that my program is thread safe. Below are specific examples of the concurrency issues I faced when making the program and the associated unique methods I used to overcome these issues:

- When 2 or more golfer objects attempt to get balls from the shared stash collection, a race condition is created. This is as 2 or more objects are attempting to read from the same memory location at the same time. This program used block synchronization to ensure that only one thread can execute the getBucketBalls method at a time.

- When Bollie takes to the field to collect the balls, mutual exclusion is required to ensure that Bollie is enabled to read from an un-adjustable collection of golf balls to ensure that each and every ball in the range is collected. My program uses an if statement to check whether or not Bollie is on the field, if so, golfer threads are required to wait (using the wait method) until such a time that Bollie is off the range. This is when the notify method will be called to continue waiting threads.

- When golfers hit a ball onto the field a race condition with regards to which thread gets to write to the collection of balls on the field. To solve this, it was necessary to use a synchronized method to add balls to this collection.

- A "ConcurrentModificationException" was given when attempting to make Bollie collect the balls from the range when golfer threads were running. This means that I had to make all golfer threads collect balls from the stash in a synchronized manner as the concurrent reading of one thread (Golfer) matched with the writing of another thread (Bollie) made the program thread unsafe. An ArrayBlockingQueue data structure was also required to store the golfballs in order for my program to work.