Homework 4 - Report

Group Members
Anton Hammar - antham@kth.se
Ludwig Flodin - ludwigfl@kth.se

The Hungry Birds Problem

The code is divided into 4 parts, producer, consumer, monitor and the main class. For this problem there are multiple consumers called baby birds and one producer called parent bird. To start off the program the thing that is to be consumed, the worm counter is set to 0 to activate the producer part of the code. This is because the producer, parent bird, is waiting for the conditional variable worms to reach 0 before it calls to restock the counter.

The multiple consumers are threads which represent baby birds, each thread will wait for the conditional variable worms to be bigger than 0, the parent bird will notify all threads when this has happened which will force the threads to start running the consume functions from their wait state. The threads will enter a FIFO queue when waiting for the conditional variable and run each code in order until the worm counter is 0, it will notify all threads when the worm variable has hit 0 which the parent bird will notice.

The restock and consume functions all happen in the monitor class and the producer and consumer class call these functions over and over again between sleep periods in an infinite while loop. The functions are synchronized and handle the multiple threads calling by placing them into a FIFO queue. The main class is used to create and start running the threads.

The Bear and Honeybees Problem

The Bear and Honeybees Problem works very similarly to The Hungry Birds Problem. It starts by creating a monitor class called *Pot* and a number of *Bee* threads which it calls .start() on to run them. Then it proceeds to create a *Bear* thread and starts that too.

The monitor class *Pot* has two synchronized methods, add() and empty().

The add method has a while loop which calls wait() if the pot is full, to prevent bees from adding honey if there is no more room in the pot. If wait() is broken it adds +1 to the honey amount in the pot and then prints the events that transpired.

The empty method has a while loop which calls wait() when the pot is not full, to prevent the bear from eating from the pot until it is filled up. After the wait() is broken it sets the honey amount to 0 and prints the transpired events.

Each bee thread has an ID which is used when printing out interesting events. The bee starts by entering an infinite while loop where it will continue to gather honey to a honeypot. It does this by calling .add() and then waiting for two seconds by calling sleep(2000). If the pot is full it will wait until it can be added as mentioned earlier when describing the add method.

The bear thread also enters an infinite while loop. Here it will call the empty() method every two seconds to try and empty the pot. However, as mentioned when describing the empty() method, it will only empty the pot when its full.

Fairness

Both the The Hungry Birds Problem and The Bear and Honeybees Problem are not “fair” with respect to both consumer and producer. In the bird problem it is unfair to the producer (the parent bird), and in the bee problem it is unfair to the consumer (the bear).