Concurrency

Franck van Breugel

March 22, 2018

1 Pot class

```
public class Pot {



  /**
   * Initializes this pot with the given size.
   *
   * @param size the number of servings this pot can contain.
   */
  public Pot(int size) {



  }

  /**
   * The cook refills this pot when the pot is empty.
   *
   * @exception InterruptedException if the cook is interrupted
   * while waiting when the pot is empty.
   */
  public void refill() throws InterruptedException {
```

```
  }

  /**
   * A savage takes a serving from this pot.
   */
  public synchronized void take() {

  }
}
```

2 Cook

```
public class Cook extends Thread {




  /**
   * Initializes this cook with the given pot.
   *
   * @param pot the pot this cook refills.
   */
  public Cook(Pot pot) {



  }

  /**
   * Refills the pot whenever it is empty until this cook is interrupted.
   */
  public void run() {









  }
}
```

3 Savage

```
public class Savage extends Thread {



  /**
   * Initializes this savage with the given pot.
   *
```

```
   * @param pot the pot this cook refills.
   */
  public Savage(Pot pot) {



  }

  /**
   * Takes a serving from the pot.
   */
  public void run() {









  }
}
```

4 DiningSavages

```
public class DiningSavages {
  public static void main(String[] args) {
    final int SIZE = 2;
    final int SAVAGES = 4;

    // create a pot


    // create a cook


    // create SAVAGES savages



    // start the cook
```

```
    // start the savages




    // wait until all savages are done (use the join method)




    // interrupt the cook

  }
}
```