**日報**

2023年08月04日

氏名：小松大二

|  |
| --- |
| ■目標  ・Problem12に取り組む  ＊エンジン等のクラスをすべて新規で作成して、そのクラスで新しくボートをつくる  ＊いずれかのボートが燃料切れを起こしたらレースを中断する処理から、全てのボートが燃　　料切れを起こしたらレースを中断する処理に修正する  ＊レースの状況を＞＞＞で視覚的に表示する |
| ■作業内容  ・Problem12に取り組む  ＊エンジン等のクラスをすべて新規で作成して、そのクラスで新しくボートをつくる  →完了  ＊いずれかのボートが燃料切れを起こしたらレースを中断する処理から、全てのボートが燃　　料切れを起こしたらレースを中断する処理に修正する　→完了  ＊レースの状況を＞＞＞で視覚的に表示する  →完了 |
| * 習得したこと   ・新しくエンジン、プロペラ、ドライバー、ボートのクラスを作って、それをもとにボートを作成しレースに参加させることができる  ・いずれかのボートが燃料切れを起こしたらレースを中断する処理を修正し、全てのボートが燃料切れを起こしたらレースを中断する処理を実装することができる  ・ int boatsOutOfFuel = 0;の書く位置によっては、処理が無限ループされてしまうことを学んだ。（最初、for内に書いてしまったことでif (list.size() == boatsOutOfFuel)条件がいつまでも満たされず無限ループが発生してしまった）  ・distanceMapの役割を理解し、ボートが進んだ分だけdistanceMapに進行距離を更新することができる  ・ StringBuilderとappendの使い方と、raceメソッド内のcurDistanceが、distanceMapからボートの進行状況を取得する役割を持っていることを理解し、その処理を真似てgraphicalRaceメソッド内に変数positionを用意し、そこから進行状況を取得してboatLineに＞＞＞を追加する処理を書くことができる  ・toStringがStringBuilderの中身を取得することができることを理解し、toStringを使ってboatLineの中身を取得して、System.out.printlnで結果を表示する処理を書くことができる。 |
| * 感想   今日は昨日に引き続きProblem12に取り組み、また修正作業にも取り組みました。まずレースの状況を視覚的に表す問題について、昨日の時点でなんとなく道筋は見えていたのですが、細かい部分の処理がいまいちピンと来ていませんでした。しかし、日報のコメントで教えていただいたヒントを参考に、raceメソッドの処理を見直したり、書いてみたりすることで、修得したことに上記したように、distanceMapの役割やcurDistanceについての理解が鮮明になりました。モヤモヤがすっきりしたので嬉しかったです。鮮明になったことで、graphicalRaceメソッド内に変数positionを用意し、そこから進行状況を取得してboatLineに＞＞＞を追加する処理を書くことができました。かなり苦戦した問題だったので、実装することができてとても嬉しかったです。  全てのボートの燃料が０になったときにレースを中断する処理を書く問題について、全部のボートの燃料が０になったことを表すにはどうしたらよいか悩みましたが、Listからボートの数を取得して、その取得した数が燃料切れを起こしたボートの数と一致したらレースを中断するようにすればいいと考えました。この処理を実装するために、燃料切れを起こしたボートを数える変数を用意しました。そして、元々書いていた、いずれかのボートが燃料切れを起こしたら中断する処理を少し書き換え、燃料切れを起こしたボートが出てきたら、さきほどの変数に追加する処理を書きました。最後に、その変数の数とListの数が一致したらレースを終える処理を書いて、無事に全てのボートが燃料切れを起こしたらレースを中断する処理を実装することができました。結果的に修正することになったけど、自分で書いた、いずれかのボートが燃料切れを起こしたらレースを中断する処理も少し活かして実装できたので、無駄にならなくて嬉しかったです。 |