ORATILE RAPOO

ASSIGNMENT 2

CSC2002S

ALL CHANGES I HAVE MADE

Added Class

HungryWordMove.java

This class is responsible for moving the HungryWord horizontal across the screen

- Instance Variables
  - FallingWord[] myWord → this stores the shared array of the hungry word
  - AtomicBoolean done → the state of the game
  - AtomicBoolean pause→ stores the state of the game
  - Score score→ stores the players current score
  - CountDownLatch startLatch→ helps this thread to start at the same time as other threads
- Methods
  - HungryWordMover(FallingWord[] word) → constructor to initialize myWord
  - HungryWordMover(FallingWord[] word,WordDictionary dict, Score score,CountDownLatch startLatch, AtomicBoolean d, AtomicBoolean p) → this constructor initializes all sate variables.
  - run() → this method is used to start the thread running , the methods start by waiting for all the threads to be ready to start, while the game is not done the ,while the hungryWord is not dropped drop the word using Hdrop(10) method from falling words and if the player does not catch the word and the word reaches the end we reset the word randomly form dictionary and increment missed counter.

Changes to existing Classes

FallingWord

This class is responsible for creating the falling word both horizontal and vertical

- Instance Variables
    - int maxX – stores the maximum width it is similar to maxY which stores maximum height.
- Methods
    - FallingWord(String text,int x, int maxY) → made the constructor take the maxY which will be used to assist in catching words so is the maxX that is initialized to default value of 880;
    - `synchronized` hSetX(int x) → this is used to set the x position of a hungryWord when its being dropped it is similar to setY(y)
    - `synchronized int getMaxY()` → gets the maxY to be used for when catching duplicate words
    - synchronized void hResetPos() → used to reset the position of the hungryword it is a helper function for hResetWord()
    - synchronized void hResetWord() → used to reset the hungryWord
    - synchronized boolean matchWord(String typedText)→ this method nologer resets the falling word we will do this manually it only checks if given word matches the hungryWord.
    - synchronized void hungryDrop(int inc) → used to drop the hungryWord by incrementing its x position.

`CatchWord`

This class is used to cathch a word

- Instance Variables
    - FallingWord[] Hword → stores the array of hungry Word which is shared among all threads
- Methods
    - run() → used to start the thread running ,in this method I used used the falling word getMaxY() method to set the initial highest caught falling word y position, if the new duplicate word caught position is below initial override loop all duplicates and choose the smallest. Use new matchWord and resetWord manually for hungryWord use hResetWord and for others use resetWord

WordMover

This class is used to move all words vertically and drop their position, it also helps in reseting words that collide with the HungryWord.

- Instance Variables
  - FallingWord[] Hword → stores the HungryWord
  - int x → stores the x position of the current falling word
- Method
  - Run() → used to start the thread , in this method I added that after dropping a word , check if the word is colliding with the HungryWord and if it collides reset the word(this will get a new word from dictionary that will start at the top of the screen) and increment missed counter else compute as normally

GamePanel

This class is responsible for painting the game panel

- Instance Variables
  - FallingWord[] Hword → stores the Hungryword
- Methods
  - GamePanel(FallingWord[] words,FallingWord[] Hword, int maxY, AtomicBoolean d, AtomicBoolean s, AtomicBoolean w) → the new constructor now takes in the array that stores the HungryWord
  - paintComponent(Graphics g)→ `this method sets the color of HungryWord green using g.setColor(Color.green) and it then draws the string`
  - synchronized public int getValidXpos()→ get valid x position is now synchronized to avoid race conditions

TypingTutorApp

This is the main class

- Instance Variables
  - FallingWord[] Hword → stores the hungry word
  - HungryWordMover hWrdShft → stores the hungry word mover object
- Methods
  - createHungryWordMoverThread() → this method creates a hungryword mover thread and starts it and it also uses a countdownlatch to wait for other threads before

- createThreads() → also called this method for the hungryWord createHungryWordMoverThread()
- main()→ in this method I initialized the hungryWord and then passed it to be caught

Thread Safety

- All the getters and setters that I created are synchronized for example the newly created methods(getMaxX() or getMaxY() or hSetX()) in FallingWord are synchronized to ensure that only one thread can use this methods at a time.
- All methods that change state variables are synchronized for example hResetWord() method which resets the x positions of HungryWord.
- Used Atomic variables for all variables that are shared among threads this ensures that all threads have the updated copy and also supports mutual exclusiveness one thread can access at time.
- Used a shared array HungryWord which is pass by reference meaning every thread had access to updated current HungryWord.
- Used a coundownLatch to make all threads to wait for another before starting this ensured all threads start at the same time when the game begins, I also used this for hungryWord

Race Conditions identified

- getValidXpos() → this method was not synchronized meaning any thread can use this method at any time since this method leading to a data race when two or more threads are reading and coping at the same time and this can also lead to bad interleaving. This caused some threads to have invalid x positions. Fix this by making the method Synchronized