```java
/**
 *
 * Author: Lukas Ottenhof
 * Date: April 15, 2023
 * Class: AUSCI 235
 *
 * this class is for a duck duck goose game. It creates a circular double linked
 * list that acts as the game circle where players are the nodes. First the it
 * player is removed from the list to simulating this player getting up, and
 * then the circle is rotated simulating the it player walking around the
 * circle saying, duck, duck, goose. once the it player picks a goose the goose
 * player is removed from the circle and they both "run" in opposite directions
 * following their path back to the starting spot. the players running is done
 * by moving a cursor through the linked list, printing a player and then
 * getting next or previous. Once a player reaches this original spot they will
 * be entered back into the main list at the spot where the goose player got up
 * and the winning player will be declared the winner. The speed at which the
 * players go is really how many spots they move around the circle before the
 * next person goes. so if the it person has speed 4 and the goose has speed 3
 * the it person goes forward 4 spots and then the goose person does 3. they
 * will take turns running like this until someone reaches where they started
 * running. if both speeds are equal the it person will go first to give them
 * a chance.
 */
import java.util.Scanner;//used to get user inputs
import java.util.Random;//used to generate random numbers

public class DuckDuckGoose {

    /**
     * in main, fist the playerCircle will be created (linked list) then the
     * user will be asked for the number of players and their names. then they
     * will be asked for the amount of rounds, then a new node containing the
     * first it players info will be made, and the first it person will be
     * removed from the circle. then it gets into the for loop. the for loop
     * runs the amount of rounds the user wants. first it prints out prompts and
     * then calls a method to do the it player walking around and gets the goose
     * person, then removes the goose person. then it uses a method to get the
     * winner and another to print who won. then the final prompts are sent out
     * using the finalState methods
     *
     * @param args
     */
    public static void main(String[] args) {
        DCircLinkList playerCircle = new DCircLinkList();//list that will
        //contain players

        Random rand = new Random();//used to make random numbers
        Scanner keyboard = new Scanner(System.in);//used to get inputs
        //from the user. Namley number of players and their names

        //adding players to the list and getting the size of the list using
        //addPlayers method
        playerCircle = addPlayers(playerCircle, keyboard);
        int numPlayers = playerCircle.getSize();

        //getting how many rounds user wants to play
        System.out.println("\nHow many rounds?");
        int numRounds = keyboard.nextInt();
        System.out.println();

        //making a node that contains the first it person. The first it person
        //is the last person entered by the user
        DNode itPerson = playerCircle.getLastNode();
        playerCircle = removeFirstItPlayer(playerCircle);

        //this for loop loops the amount of round that the user wants to play.
        //1 loop is one round
        for (int round = 0; round < numRounds; round++) {
            System.out.println("+++++++++++++++++++++++++++++"
                    + "+++++++++++++++\n");
            System.out.println("Round " + (round + 1));

            System.out.println("Game circle:   " + playerCircle.toString());
            System.out.println("It-Person:   " + itPerson.getElement());

            playerCircle = itPersonWalking(playerCircle, rand); //it person on
            // the outside walking and picking a goose

            //making a person node that is the same as the node containing the
            //goose person
            DNode goosePerson = playerCircle.getFirstNode();

            //removing the goose person from the circle
            playerCircle.removeFirstNode();

            System.out.println("Game circle:   " + playerCircle.toString());

            //gettting our winner
            String winner = theRace(playerCircle,
                    numPlayers, rand, itPerson, goosePerson);

            //this adds the winner back into the list
            playerCircle = addWinnerToList(playerCircle, itPerson, goosePerson,
                    winner);

        }//end of round loop (for loop)

        //final prompts
        finalState(playerCircle, itPerson);
    }//end of main

    //methods==============================================================
    /**
     * this is for the start of the game, it asks the user for how many people
     * are playing and what their names are. all of this is stored in the
     * playerCircle linked list using the addLast method.
     *
     * @param playerCircle this is the linked list that i will be building onto.
     * it will store the players names
     *
     * @param keyboard this is used to get the users input on how many people
     * there are. it is taken as a parameter rather than having it put in here
     * because the scanner is used in other methods and this way i wont have to
     * make a scanner more than once
     *
     * @return playerCircle will be returned. this will be the updated version
     * of the list that now contains the players and their names
     */
    public static DCircLinkList addPlayers(DCircLinkList playerCircle,
            Scanner keyboard) {

        System.out.println("How many players?");
        int numPlayers = keyboard.nextInt();

        //this will loop once per number of players wanted by user
        for (int playerCount = 0; playerCount < numPlayers; playerCount++) {
            System.out.println("Please enter player " + (playerCount + 1)
                + "'s name:");
            String playerName = keyboard.next();
            playerCircle.addLast(playerName);
        }//end of for loop
        return playerCircle;
    }//end of addPlayers

    /**
     * this method is used to remove the first "it" player at the start of the
     * game. it is only used once, it does this by rotating the list so the last
     * person entered if first, then this node is removed
     *
     * @param playerCircle this is the game circle. one player will be removed
     *
     * @return the updated version of the list will be returned
     */
    public static DCircLinkList removeFirstItPlayer(DCircLinkList playerCircle){
        for (int i = 0; i < playerCircle.getSize() - 1; i++) {
            playerCircle.rotateCCW();
        }

        playerCircle.removeFirstNode();
        return playerCircle;
    }//end of remove first it player

    /**
     * this method is used when the it person is walking around the outside of
     * the circle saying "duck duck duck ..... goose!" it generates a random
     * number between 1 and 20 and this is how many ducks there will be before
     * the goose. it then rotates the list says duck and then the name of the
     * new person that is at the start after the rotations are done.
     *
     * @param playerCircle this is the list that i had put all the players into.
     * with each duck the list will be rotated.
     *
     * @param rand used to make the random number between 1 and 20, this is to
     * pick how many people are ducks before a goose is picked
     *
     * @return the updated version of the list that has been rotated will be
     * returned
     */
    public static DCircLinkList itPersonWalking(DCircLinkList playerCircle,
            Random rand) {
        System.out.println("=======================================");
        int numOfDucks = rand.nextInt(20) + 1;//this will generate a random
        //number between 0 and 19 and then add 1 to it so we get a random number
        //between 1 and 20

        System.out.println("Random number generated is: " + numOfDucks);

        //this will loop the amount of times that our random number indicates,
        //rotating the list and saying the new fisrt players name each time
        for (int duckCount = 0; duckCount < numOfDucks; duckCount++) {
            System.out.print(playerCircle.getFirstElement());
            System.out.print(" Duck; ");
            playerCircle.rotateCCW();

        }//end of duck loop

        //after all of the ducks have been said, the goose will be declared
        System.out.print(playerCircle.getFirstElement());
        System.out.println(" GOOSE \n");
        System.out.println("Up jumps:   " + playerCircle.getFirstElement());

        return playerCircle;//returning the updated list
```

```java
    }//end of itPersonWalking

    /**
     * This method is for the main part of the game when the it player and the
     * goose player are racing back to their original spot. this works by
     * generating random numbers for speed, this is how many spots a player will
     * run past before the other player runs, and the player with the higher
     * speed runs first. this method uses the run method to do the running part
     * which takes the players cursor and moves it the amount of times their
     * speed is equal to, and prints each player they pass. it checks to see if
     * there is a winner by checking to see if the cursor is null after their
     * run, if it is null this is the way im indicating there is nobody else to
     * pass
     *
     *
     * @param playerCircle this is the list containing the players
     * @param numPlayers number of players in the game
     * @param rand to make random ints for a players speed
     * @param itPerson the person who is it
     * @param goosePerson the person who is the goose
     * @return the winning person will be returned
     */
    public static String theRace(DCircLinkList playerCircle, int numPlayers,
            Random rand, DNode itPerson, DNode goosePerson) {

        String winner = ""; //the string that will be returned

        //these are the total numbers of people who the players have passed
        int goosePassed = 0;
        int itPassed = 0;

        //making the cursors that will go through the list for each player. the
        //goose cursor is set to get previous becasuse the goose is runnning the
        //other direction and there for the first person they pass is different
        DNode gooseCursor = playerCircle.getFirstNode();
        playerCircle.setEntry(gooseCursor);
        gooseCursor = gooseCursor.getPrevious();

        DNode itCursor = playerCircle.getFirstNode();
        playerCircle.setEntry(itCursor);

        //this loop will continue until a winner has been returned. it calls
        //the players turn method until there is a winner
        while (winner.equals("")) {
            //cursors for going through the list

            //this will generate a random speed for the player between 1 and the
            //number of players
            int gooseSpeed = rand.nextInt(numPlayers - 2) + 1;
            int itPersonSpeed = rand.nextInt(numPlayers - 2) + 1;

            System.out.println("   **Speeds:   It-Person " + itPersonSpeed +
                    " Goose " + gooseSpeed);

            if (gooseSpeed > itPersonSpeed) {// if goose is faster

                //this will update the goose cursor and use the person running
                //method
                gooseCursor = personRunning(playerCircle, gooseCursor,
                        gooseSpeed, numPlayers, goosePassed,
                        "goose");
                //if the cursor was returned as null this person won and the
                //loop will end
                if (gooseCursor == null) {
                    winner = "goose";
                    return winner;
                }

                //speed is added to the total passed becasue the speed is how
                //many players the player passed this round. this is the reason
                //for all times speed is added to total
                goosePassed += gooseSpeed;

                //this will update the goose cursor and use the person running
                //method
                itCursor = personRunning(playerCircle, itCursor,
                        itPersonSpeed, numPlayers, itPassed,
                        "it");
                //if the cursor was returned as null this person won and the
                //loop will end
                if (itCursor == null) {
                    winner = "it";
                    return winner;
                }

                itPassed += itPersonSpeed;
                System.out.println();
            }//end of else goose is faster
            else {//if it person is faster

                //this will update the goose cursor and use the person running
                //method
                itCursor = personRunning(playerCircle, itCursor,
                        itPersonSpeed, numPlayers, itPassed,
                        "it");
                //if the cursor was returned as null this person won and the
                //loop will end
                if (itCursor == null) {
                    winner = "it";
                    return winner;
                }

                itPassed += itPersonSpeed;

                //this will update the goose cursor and use the person running
                //method
                gooseCursor = personRunning(playerCircle, gooseCursor,
                        gooseSpeed, numPlayers, goosePassed,
                        "goose");
                //if the cursor was returned as null this person won and the
                //loop will end
                if (gooseCursor == null) {
                    winner = "goose";
                    return winner;
                }

                goosePassed += gooseSpeed;
                System.out.println();

            }//if it person is faster

        }//end of while
        return winner;//this should never actualy be returned
    }//end of theRace

    /**
     * this method runs for each persons run portion of the race.this method
     * moves the players cursor the amount of times that their speed corresponds
     * to. once the player has done a full loop their path will be set to null
     * so the method this one is within knows that the player completed their
```

```
     * path and there is no more players for them to pass. it is used for both
     * players, and also sends out a prompt telling the user who is running
     *
     * @param playerCircle the list containing the players
     * @param cursor the cursor of the player running
     * @param personSpeed the players speed (how many times the list will be
     * rotated)
     * @param numPlayers number of players in the game
     * @param peoplePassed how many people the player has run past. you know a
     * player has run a full loop when the num of people they have passed is
     * equal to the number of people in the loop
     * @param player tells us which player is running so we know which direction
     * to turn the list
     * @return the updated version of the players path will be returns
     */
    public static DNode personRunning(DCircLinkList playerCircle, DNode cursor,
            int personSpeed, int numPlayers, int peoplePassed, String player) {

        if (player.equals("it")) {
            System.out.print("      It-Person running past: ");
        } else {
            System.out.print("      Goose running past:     ");
        }
        for (int i = 0; i < personSpeed; i++) {

            peoplePassed++;

            //if the people passed is equal to the num of players in the circle,
            //the runner has passed everyone and has won
            if (peoplePassed == numPlayers - 2) {
                System.out.print(cursor.getElement() + " ");
                return null;
            }//end of if
            //otherwise the player being passed will be printed and the cursor
            //will be moved
            else {

                if (player.equals("it")) {
                    System.out.print(cursor.getElement() + " ");
                    cursor = cursor.getNext(); //b

                } else {
                    System.out.print(cursor.getElement() + " ");
                    cursor = cursor.getPrevious(); //b
                }

            }//end of else

        }//end of it turn
        System.out.println();

        return cursor; //returning the updated cursor
    }//end of personRunning

    /**
     * this is the method that adds the winning player back into the list and
     * prints out who won to the user. it knows who won by the param winner
     *
     * @param playerCircle this is the list containing the player sitting
     * @param itPerson this is the nod containing the it persons name
     * @param goosePerson this is the nod containing the goose persons name
     * @param winner this is an String that indicates who won
     * @return
     */
    public static DCircLinkList addWinnerToList(DCircLinkList playerCircle,
            DNode itPerson, DNode goosePerson, String winner) {

        //this is the node that will be added to the list. it will be
        //updated to contain the name of the winner
        DNode thisPersonWon = new DNode();

        //putting the winner back into the list (the circle)
        if (winner.equals("it")) {
            thisPersonWon.setElement(itPerson.getElement());
            playerCircle.addNodeAsFirst(thisPersonWon);
            System.out.println("\n\nIt-Person (" +
                    itPerson.getElement() + ") wins");
        } else {
            thisPersonWon.setElement(goosePerson.getElement());
            playerCircle.addNodeAsFirst(thisPersonWon);
            System.out.println("\n\nGoose ("
                    + goosePerson.getElement() + ") wins");
        }

        //if the it person won the goose is the new it person. if goose won
        //there is no need for change
        if (winner.equals("it")) {
            itPerson.setElement(goosePerson.getElement());
        }

        return playerCircle;
    }// end of addWinnerToList


    /**
     * this method is for the end of the game. It display the final it person
     * and the final game circle
     *
     * @param playerCircle the final game circle
     * @param itPerson the last it person
     */
    public static void finalState(DCircLinkList playerCircle, DNode itPerson) {
        System.out.println("+++++++++++++++++++++++++++++"
                + "+++++++++++++++\n");
        System.out.println("Done. Final State.");
        System.out.println("Game circle: " + playerCircle.toString());
        System.out.println("It-Person: " + itPerson.getElement());
        System.out.println("========================================");
    }//end of final state

}//end of class
```