Progetto: FORZA 4(protocollo)

A cura di Luca Song, Radoslaw Jakub Zak

Introduzione:

Progettazione del gioco "Forza 4", mediante l'uso di socket dei server concorrenti, scritto in linguaggio java.

Lato client:

il client creerà il socket, per poter mandare il dati, dopo aver ricevuto sia la porta in cui entrerà, sia l'indirizzo necessario, e controllerà se gli utenti abbiano effettuato le mosse per poi essere mandate al server, per poter aggiornare e rimandare al client, ove verranno visualizzate dagli utenti col GUI(visibile sulla tabella)

(quando verrà creato una nuova istanza del client, cercherà di connettersi al server di riferimento, per poter connettersi all'altro host)

```
public void client(String indirizzo,int port) {
    try {
        socket = new Socket(indirizzo, port);
        outputStream = socket.getOutputStream();
        objectOutputStream = new ObjectOutputStream(outputStream);
        System.out.println("funziona il socket!");
        executorService.submit(this::run);
    } catch (IOException e) {
        client(indirizzo,port+1);
        e.printStackTrace();
    }
}
```

(qui permette di inviare gli eventuali dati che il giocatore abbia interagito con il gioco, e spedire al server)

```
public void invioDati(Dati dato) {
    try {
        objectOutputStream.writeObject(dato);
    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

(qui il buffer viene costantemente aggiornato, prendendo le mosse dei giocatori per poi essere mandati tramite il client verso il server, ove rileverà la modifica e modificherà il vecchio contenuto, per poi essere rimandato a tutti gli host e verrà visualizzato graficamente dal gioco)

```
    public void aggionaBuffer(Dati dato1){
        for (int j = 0; j < mosse.length; j++) {
            String controllo[] = mosse[j].split("-");
            int integer = Integer.parseInt(controllo[0]); // %10 = colonne,/10= righe
            String controllo1[] = dato1.getMossa().split("-");
            int integer1 = Integer.parseInt(controllo[0]);
            if (integer % 10 == integer1 % 10 && integer / 10 == integer1 / 10 && controllo[1].equals("n") == true) {
                String buffer1[] = mosse;
                for (int i = 0; i < buffer1.length; i++) {
                    if(i==j){
                        buffer1[j] = dato1.getMossa();
                    }
                }
                break;
            }
        }
    }
}
```

Lato Server:

quando viene inizializzato il due server(uno per ciascun host), viene creata anche l'istanza della classe BufferServer.java, poiché tra i due host verranno condivisi un buffer che possa contenere tutte le mosse commessi dai giocatori.

Una volta che si viene create le due istanze del server, si cerca di ottenere gli indirizzi per poter connettersi, e allo stesso tempo cerca di eseguire il due metodi, chiamati run() e contrBufferServerCondiviso(), in cui lo scopo è di ottenere il dati inviati dal server e allo stesso tempo di controllare le variazioni tra il buffer condiviso, e il buffer locale, se riscontra le variazioni, allora manda l'oggetto "dato"(che conterrà la variazione e verrà modificata il contenuto del buffer condiviso).

(quando vengono inizializzati il due server concorrenti per gli host, verrà fatto passare il buffer condiviso, e il ID relativo)

```
public Server(BufferServer buffer, int num) {
    try {
        this.buffer = buffer;
        executorService = Executors.newFixedThreadPool(2);
        server = new ServerSocket(num,0);
        server.setReuseAddress(true);
        System.out.println("Server" + num + " attivo " + server.getInetAddress().getLocalHost());
        richiestaClient = server.accept();
        System.out.println("connesso con " + richiestaClient + "!");
        outputStream = richiestaClient.getOutputStream();
        objectOutputStream = new ObjectOutputStream(outputStream);
        invioDati1();
        this.start();
    } catch (IOException e) {
        server(buffer, num + 1);
        e.printStackTrace();
    }
}

public void server(BufferServer buffer, int num) {
    try {
        this.buffer = buffer;
        executorService = Executors.newFixedThreadPool(2);
        server = new ServerSocket(num);
        System.out.println("Server" + num + " attivo " + server.getInetAddress().getLocalHost());
        richiestaClient = server.accept();
        System.out.println("connesso con " + richiestaClient + "!");
        outputStream = richiestaClient.getOutputStream();
        invioDati1();
        executorService.submit(this::contrBufferServerCondiviso);
        executorService.submit(this::run);
    } catch (IOException e) {
        server(buffer, num + 1);
        e.printStackTrace();
    }
}
```

(quando si crea il due server, controlleranno se dal lato client riceve dati, nel tempo reale)

```java
public void run() {
    try {

        inputStream = richiestaClient.getInputStream();
        objectInputStream = new ObjectInputStream(inputStream);
        while (true) {
            dato = (Dati) objectInputStream.readObject();
            Dati dato1 = new Dati(dato.getMossa(), "server", dato.getOp());
            invioDatiGenerico(dato1);
            Thread.sleep(250);
        }
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

public void contrBufferServerCondiviso() {
    Dati dato1;
    try {
        while (true) {
            if (!bufferLocal.getmosse().equals(buffer.getmosse())) {
                String mosse1[] = buffer.getmosse();
                for (int i = 0; i < mosse1.length; i++) {
                    if (!mosse1[i].equals(bufferLocal.getmosse()[i])) {
                        invioDatiGenerico(dato1 = new Dati(mosse1[i], "server", Dati.Operazione.MOS));
                    }
                }

            }

            Thread.sleep(250);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
```

(dopo che viene confrontato il buffer condiviso con quello modificato, viene automaticamente modificato il contenuto del buffer condiviso con quello modificato dal client)

```
public void aggiornamentoBuffer(Dati dato) {
    for (int j = 0; j < buffer.getmosse().length; j++) {
        String controllo[] = buffer.getmosse()[j].split("-");
        int integer = Integer.parseInt(controllo[0]); // %10 = colonne,/10= righe
        String controllo1[] = dato.getMossa().split("-");
        int integer1 = Integer.parseInt(controllo[0]);
        if (integer % 10 == integer1 % 10 && integer / 10 == integer1 / 10 && controllo[1].equals("n") == true) {
            String buffer1[] = buffer.getmosse();
            buffer1[j] = dato.getMossa();
            bufferLocal.setmosse(buffer1);
            buffer.setmosse(buffer1);
            break;
        }
    }
    System.out.println(buffer.toString());

}
```

(inviodati1() :nel caso che venisse inizializzato il due server, allora comunica al client il dati legato al singolo giocatore, se è giocatore rosso o giallo)

(inviodatiGenerico() :nel caso invece che uno dei giocatori dovesse fare la sua mossa, allora verrà mandata il dati che modificheranno il contenuto del buffer tra il due server, per poi essere visualizzato dagli utenti tramite GUI, altrimenti non verrà fatto nessuna modifica)

```
public synchronized void invioDati1() {
    try {
        objectOutputStream.writeObject(dato = new Dati("server", Dati.Operazione.NOM, controlloUsers()));
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public synchronized void invioDatiGenerico(Dati dato) {
    try {
        objectOutputStream.writeObject(dato);
        System.out.println("mandato mesaggio\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```