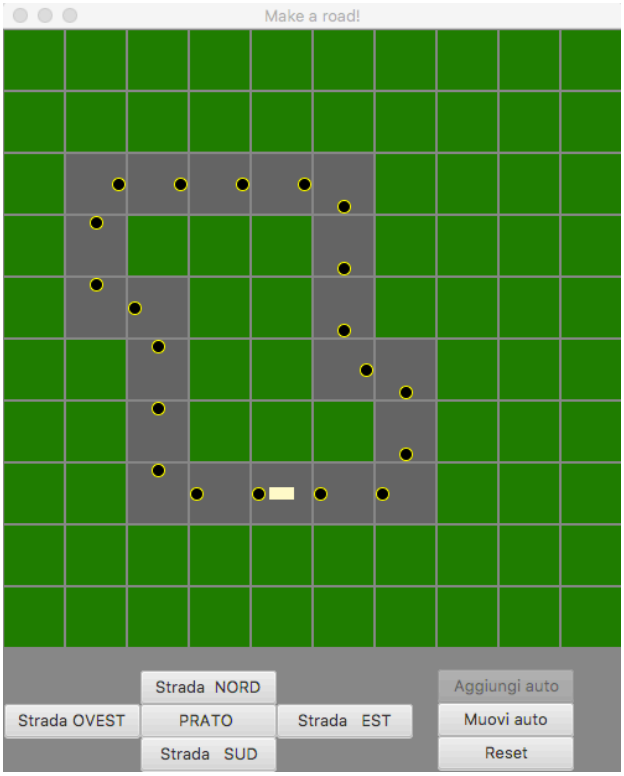
Compito di
Linguaggi di Programmazione – Programmazione 2 – prof. Picco
Linguaggi di Programmazione (Mod.1) - prof. Ronchetti

Prova al calcolatore – 9 gennaio 2020

1) E richiesto il class diagram UML. È sufficiente la vista di alto livello (senza attributi/metodi) ma devono essere mostrate le associazioni più importanti, oltre a quella di ereditarietà. Il diagramma UML deve essere consegnato su un foglio protocollo indicando nome, cognome, numero di matricola. Si prepari su un foglio il diagramma UML di tutte le classi coinvolte nel progetto sotto specificato. Non serve mostrare le classi di JavaFX, a parte quelle che hanno un ruolo diretto nel programma, e quelle da cui le classi scritte dal programmatore ereditano direttamente, se ve ne sono. Al termine, il foglio (sul quale devono essere presenti nome, cognome, numero di matricola) dovrà essere consegnato.

2) Si costruisca una interfaccia composta di una griglia quadrata (lato 500 px) di 10x10 celle e due gruppi di bottoni, possibilmente disposti come in figura e descritti ai successivi punti 5 e 7. La dimensione totale della finestra è 500x600.



3) Le celle della griglia possono essere di vario tipo: “Prato” (verdi) o “Strada” (grigie). Queste ultime si suddividono in quattro categorie: “Nord”, “Sud”, “Est”, “Ovest”, caratterizzate da un piccolo cerchio posto rispettivamente in alto, in basso, a destra e a sinistra.

4) Inizialmente la griglia è interamente composta di celle di tipo “Prato”.

5) Il primo gruppo di bottoni comprende quelli che descrivono il tipo di cella: “Nord”, “Sud”, “Est”, “Ovest”, “Prato”. Questi bottoni sono mutuamente esclusivi, ovvero premendone uno questo si disabilita e i restanti quattro sono abilitati.

6) Cliccando sulla griglia, viene effettuata una azione che corrisponde al bottone relativo al tipo di cella attualmente disabilitato: la cella su cui si è cliccato viene sostituita con una cella del tipo indicato (ad esempio inizialmente si potrà sostituire una delle celle “Prato” preesistenti con una cella di tipo “Strada Nord”).

7) La pressione dei tasti “N”, “S”, “E”, “O”, “P” sulla tastiera è equivalente rispettivamente alla pressione dei bottoni “Nord”, “Sud”, “Est”, “Ovest”, “Prato”.

8) Nel secondo gruppo di bottoni troviamo: “Aggiungi Auto”, “Muovi Auto”, “Reset”. Il bottone “Muovi Auto” è inizialmente disabilitato.

9) Dopo aver premuto il bottone “Aggiungi auto”, il click su una cella tenterà di aggiungere una auto (rappresentabile con un rettangolo di dimensione minore della cella).
    a. Se la cella su cui si è premuto è di tipo “Prato”, un messaggio in console (o in una finestra di pop-up) dirà “la macchina non può essere aggiunta su un prato!”
    b. Se la cella su cui si è premuto è di tipo “Strada”, l’automobile verrà aggiunta su di essa e il bottone “Aggiungi auto” si disabiliterà, mentre il bottone “Muovi Auto” viene abilitato.

10) La pressione del tasto “Muovi Auto” tenterà di muovere l’automobile sulla cella adiacente nella direzione indicata dal cerchietto della cella su cui si trova (in alto, in basso, a destra o a sinistra)”
    a. Se la cella di destinazione è di tipo “Strada”, l’operazione avrà successo e la macchina verrà spostata.
    b. Se la cella di destinazione è di tipo “Prato”, un messaggio in console o in una finestra di pop-up dirà “Crash!” e la macchina resterà dove si trova.
    c. Altrettanto accadrà se la destinazione è fuori dalla griglia.

11) Il bottone “Reset” elimina l’automobile, riabilita il bottone “Aggiungi Auto” e disabilita il bottone “Muovi Auto”.

12) Si documenti il codice prodotto con Javadoc (solo studenti del prof. Ronchetti).

Altri requisiti

- Il codice generato deve rispettare i principi della programmazione object-oriented. In particolare, si usi, quando evidente/utile, una gerarchia di ereditarietà, sfruttando ove possibile il polimorfismo.
- Si usino costanti ove ragionevole, e si badi alla pulizia del codice (es., linee guida Java) evitando duplicazioni e codice inutilmente complesso.

Note

- Per raggiungere la sufficienza basta fare bene i punti da 1 a 7.
- Non è richiesto di gestire il resize della finestra.

Suggerimenti

1) Per realizzare una componente che abbia sfondo grigio e mostri un cerchio a destra, si suggerisce di usare il seguente frammento di codice:

```
int radius=10;
bp.setStyle("-fx-background-color: #666666;");
circle.setRadius(radius);
circle.setStroke(Color.YELLOW);
bp.setAlignment(circle,Pos.CENTER);
bp.setRight(circle);
```

dove **bp** è un **BorderPane** e **circle** è un'istanza di **Circle**.

2) Volendo avere i bordini delle celle, si consiglia di usare il metodo **setMargin** presente nella maggior parte dei **Pane** (**HBox**, **VBox**, **BorderPane**, **StackPane**…) e documentato nelle relative API.