Esercizio C++

Classi e polimorfismo

Esercizio 1

Si scriva il codice di una classe C++ **P** che implementi un componente di un navigatore satellitare che calcola il tempo di percorrenza, e che mantenga l'informazione sulla velocità media, che deve essere un valore **maggiore** di zero.

Il costruttore di P prende in ingresso la velocità media. Se non passata, essa deve essere impostata a 60 Km/h.

Nella classe P si devono prevedere due operazioni: una di calcolo che accetta come parametro la distanza da percorrere in km e restituisce il tempo di percorrenza; una di modifica che permetta di impostare una nuova velocità media, sempre facendo un controllo che la nuova velocità sia conforme alle specifiche.

Opzionale:

Si faccia overload dell'operatore "<<" che, chiamando toString, stampi per esempio:

"Velocità media 60 km/h"

Esercizio 2

Si scriva il codice di una classe C++ **PF** che **estenda P**, e che implementi un componente di un navigatore satellitare che calcola il tempo di percorrenza tenendo conto del traffico;

PF deve mantenere una informazione che rappresenta, in percentuale il tempo ulteriore che serve per percorrere una distanza, a causa del traffico; tale valore **deve essere non negativo** ed espresso come fattore percentuale frazionario (e.g., 0.5 per indicare il 50%).

Si preveda un costruttore che inizializzi le istanze di PF con il valore della velocità media e dell percentuale di traffico passati come parametro, sempre controllando che sia un valore corretto. Tale costruttore inizializza le istanze di PF con i valori di default di 60 km/h e il valore del tempo ulteriore corrispondente all'assenza di traffico.

Rispetto alla superclasse, **si deve sovrascrivere l'operazione di calcolo** per restituire il tempo di percorrenza incrementato, in percentuale, dal tempo ulteriore dovuto al traffico.

Opzionale:

Si faccia override di toString, affinché "<<" stampi per esempio:

"Velocità media 100 km/h percentuale traffico 10 percento"

Esercizio 3

Si faccia un main di prova in cui si mostra un comportamento **polimorfico dinamico**.

Inoltre,

Si dimostri un comportamento **polimorfico statico**. Per fare ciò, s'implementi la funzione getTempoPolyStatic che **templetizzata** prenda in ingresso un generico oggetto di classe P o PF ed una distanza e restituisca il tempo di percorrenza chiamando il rispettivo metodo all'interno dell'oggetto.

Anche questa funzione deve essere chiamata dal main.