ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсеци за СИ, ИР, ОС и ОФ Београд 19.04.2020.

Други домаћи задатак из
Објектно оријентисаног програмирања 2

1) Саставити на језику *Java* следећи пакет класа:

- Тркачко ***возило*** се ствара са задатом максималном реалном брзином коју може да достигне (*m/s*), реалним убрзањем (m/s^2), реалном управљивошћу на скали од 0 до 1 и именом такмичара који га вози. Возило садржи и реалну тренутну брзину коју развија (при стварању је 0) и она не сме да буде већа од максималне брзине. Сви атрибути возила могу да се дохвате и поставе. При постављању максималне брзине по потреби ажурирати и тренутну брзину уколико је била већа од новог максимума. Могуће је симулирати померање возила на основу задатог протеклог времена при чему је резултат операције пут који би возило прешло у метрима, а као бочни ефекат се ажурира тренутна брзина возила по формулама у наставку. Могуће је израчунати време у секундама потребно да би се прешао задати пут у метрима. При рачунању сматрати да се возило креће равномерно убрзано до достизања максималне брзине након чега наставља да се креће равномерно без убрзања. Може да се састави текстуални опис у облику *име* **[***максимална_брзина, убрзање, лакоћа_управљања***]**.

$$s = V_0 t + \frac{at^2}{2}; V^2 = {V_0}^2 + 2as;\ V = V_0 + at$$

- ***Специфичност*** садржи аутоматски генерисан јединствен целобројни идентификатор који може да се дохвати. Специфичност може да испољи или поништи ефекат над задатим објектом са могућношћу пријављивања грешке уколико задати објекат није одговарајућ (*GNeodgovarajuciObjekat*).
- ***Кривина*** је специфичност која се ствара са задатом максималном брзином коју возило сме да развије и може да се дохвати. Ефекат кривине може да се испољи и поништи искључиво над задатим возилом. При испољавању ефекта возилу се поставља максимална брзина на максималну брзину прописану кривином помножену управљивошћу возила уколико нова максимална брзина није већа од оригиналне. При поништавању ефекта максимална брзина возила се враћа на оригиналну вредност. Кривина може да се клонира. Текстуални опис кривине је облика **K***макс_брзина*.
- ***Деоница*** пута се ствара са задатом реалном дужином у метрима која може да се дохвати. Деоница пута садржи произвољан број специфичности. При стварању деоница нема ниједну специфичност након чега се специфичности могу појединачно додавати на крај. Могуће је уклонити специфичност на основу задатог идентификатора при чему је операција без ефекта уколико специфичност са задатим идентификатором не постоји. Могуће је дохватити специфичност на задатој позицији. Могуће је дохватити број специфичности. Деоница пута може да се клонира. Текстуални опис деонице у облику **`deonica(`***дужина***`m)`**, након чега се у наставку исписују све специфичности.

Приложена је класа са главном функцијом која испитује основне функционалности пакета класа уз исписивање резултата на стандардном излазу (конзоли).

НАПОМЕНЕ:

а) Други домаћи задатак је основа за израду друге лабораторијске вежбе.

б) Студент треба да преда своја решења, сходно упутствима које добије преко мејлинг листе предмета. Предата решења биће доступна студенту и користиће их као полазну тачку за израду лабораторијске вежбе.

в) Решење домаћег задатка се не оцењује, али улази у састав решења лабораторијске вежбе које се оцењује

```
================================================================================
Imena klasa i metoda navedenih u main metodi ne smeju se menjati!
================================================================================
```

```java
package main;
import karting.*;

public class Main {
   public static void main(String[] args) {
      Vozilo v1 = new Vozilo(30.0, 3.0, 0.75, "Crash Bandicoot");
      Vozilo v2 = new Vozilo(40.0, 4.0, 0.5, "Coco Bandicoot");
      v2.postMaksBrzinu(40.0);
      v2.postUbrzanje(4.0);
      v2.postUpravljivost(0.5);
      v2.postTrenBrzinu(0.0);

      System.out.println(v1.toString());
      System.out.println(v2.dohvIme() + " [" +
            v2.dohvMaksBrzinu() + ", " +
            v2.dohvUbrzanje() + ", " +
            v2.dohvUpravljivost() + "]");

      System.out.println("Crash - predjeni put za 1s: " + v1.pomeriVozilo(1) + "s");
      System.out.println("Crash - trenutna brzina: " + v1.dohvTrenBrzinu() + "m/s");
      System.out.println("Coco - vreme za 200m: " + v2.izracunajVreme(200) + "m");
      System.out.println("Coco - trenutna brzina: " + v2.dohvTrenBrzinu() + "m/s");

      Vozilo v3 = new Vozilo(45, 2.5, 0.25, "Tiny Tiger");
      Specificnost s1 = new Krivina(40.0);
      try {
         s1.ispoljiEfekat(v3);
         System.out.println("Tiny - maksimalna brzina: " + v3.dohvMaksBrzinu() + "m/s");
         s1.ponistiEfekat(v3);
         System.out.println("Tiny - maksimalna brzina: " + v3.dohvMaksBrzinu() + "m/s");
      } catch (GNeodgovarajuciObjekat e) {
         System.err.println(e);
      }

      Deonica d = new Deonica(100.0);
      d.dodajSpecificnost(s1);
      d.dodajSpecificnost(((Krivina)s1).clone());
      d.dodajSpecificnost(((Krivina)s1).clone());
      d.izbaciSpecificnost(s1.dohvatiId());
      System.out.println("Specificnost na prvoj poziciji: " + d.dohvSpecificnost(0));
      System.out.println(d);
   }
}
```

```
===============
Primer izlaza:
===============
Crash Bandicoot [30.0, 3.0, 0.75]
Coco Bandicoot [40.0, 4.0, 0.5]
Crash - predjeni put za 1s: 1.5s
Crash - trenutna brzina: 3.0m/s
Coco - vreme za 200m: 10.0m
Coco - trenutna brzina: 0.0m/s
Tiny - maksimalna brzina: 10.0m/s
Tiny - maksimalna brzina: 45.0m/s
Specificnost na prvoj poziciji: K40.0
deonica(100.0m)K40.0K40.0
```