Academia Tehnică Militară "Ferdinand I“
Facultatea de Sisteme Informatice şi Securitate Cibernetică

Paradigme de programare (în Java)

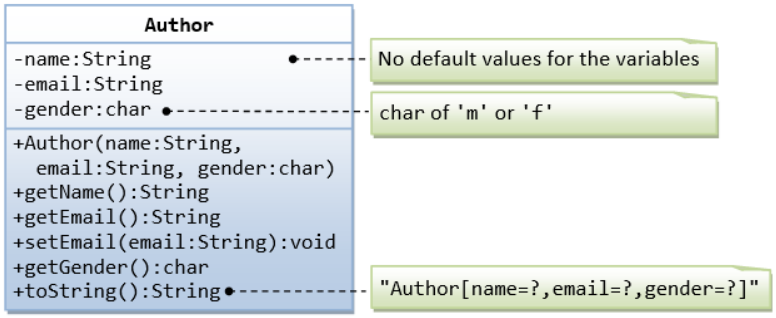
Lab 3/Curs 3- Programare orientată pe obiecte (OOP)

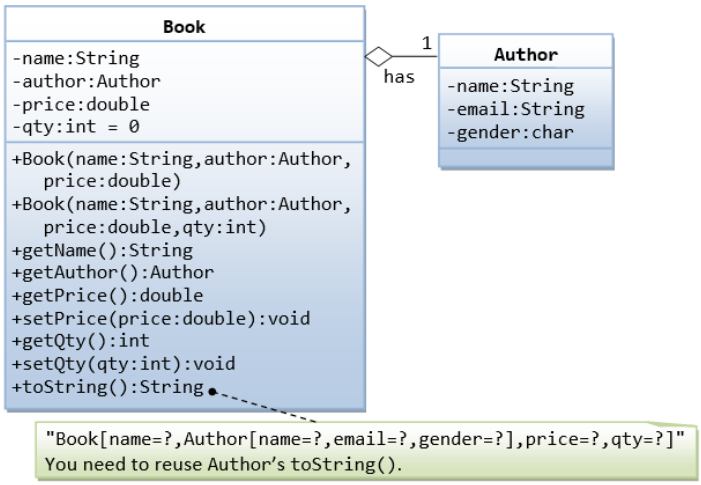
Col(r) Traian Nicula

Se creează un proiect Maven cu numele ***lab3*** și se salvează în folder-ul cu numele studentului. Pentru fiecare exercițiu se va crea câte o clasă de test (Exercise1 etc.) cu metoda statică *main*, precum și alte clase cerute de exerciții.

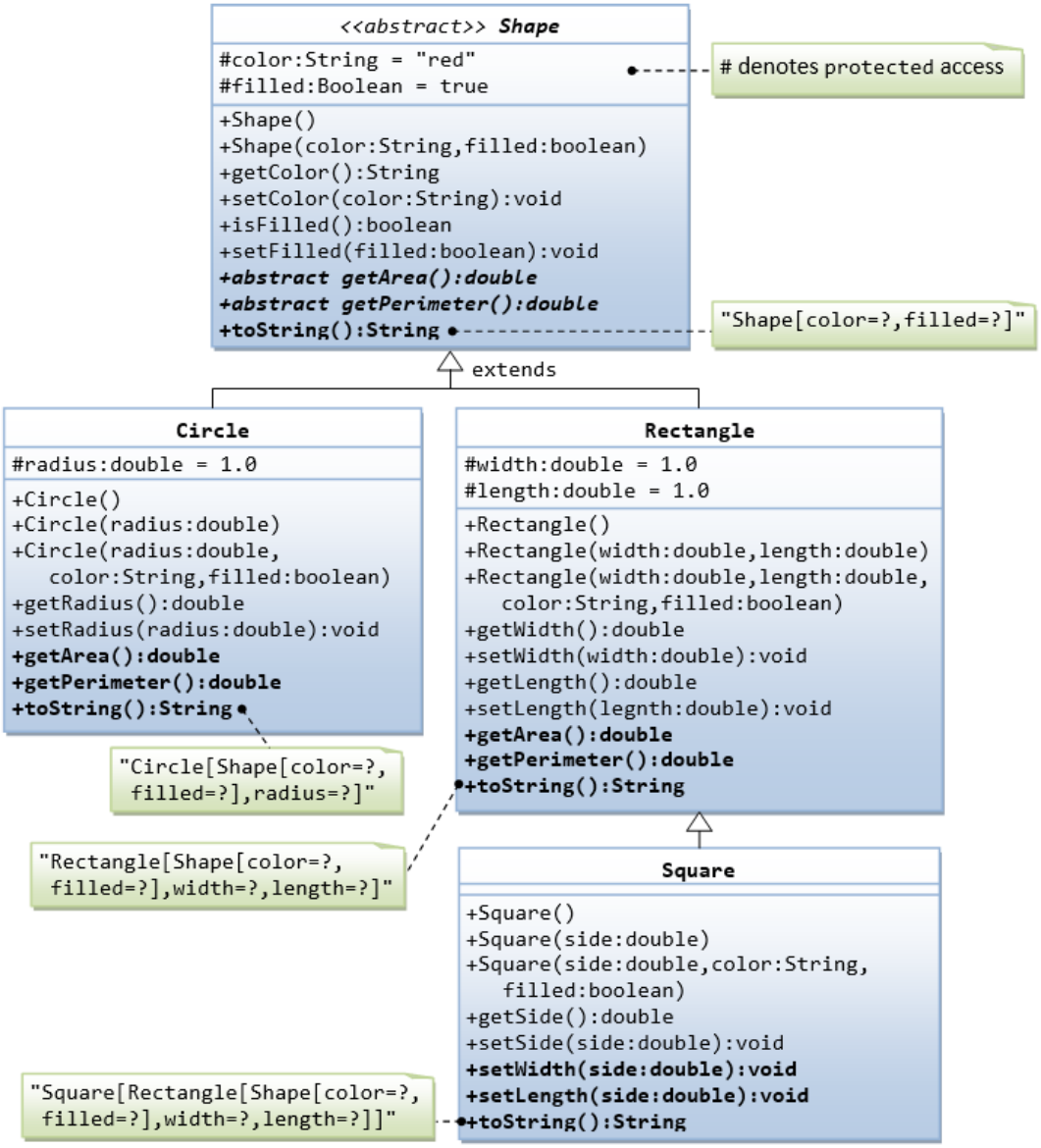
Se rezolvă următoarele exerciții:

1. Scrieți un program Java care implementează și testează (cu clasa Exercise1 cu metoda *main*) clasele definite de diagrama UML de mai jos, astfel: (**4p**)
   a. Se implementează clasele ***Author*** și ***Book*** conform diagramelor UML de mai jos (**2p**)
   b. În metoda statică main() a clasei Exercise1, se creează: (**2p**)
      - 2 instanțe pentru clasa ***Author***
      - 2 instanțe pentru clasa ***Book*** folosind ambii constructori implementați. Pentru primul constructor se setează cantitatea cu metoda setter corespunzătoare
      - se tipăresc obiectele **Book** create





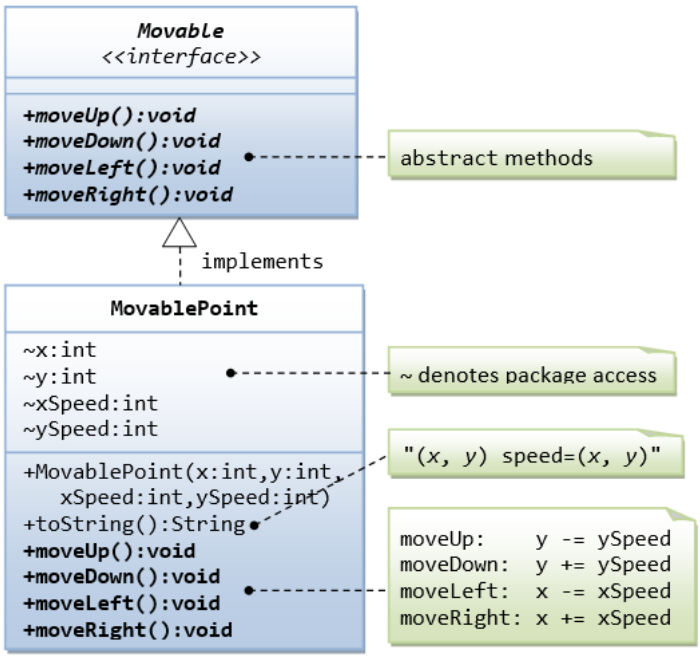
2. Scrieți un program Java cu blocuri ***try-catch*** care: (**1p**)
   - declară în ***try*** un vector cu 5 numere întregi
   - inițializează elementul 6 cu valoarea 6
   - prinde în blocul ***catch*** excepția *ArrayIndexOutOfBoundsException* și afișează mesajul de eroare.
3. Scrieți un program Java care implementează și testează clasele definite de diagrama UML de mai jos, astfel: (**4p**)
   a. Se implementează clasele ***Shape***, ***Circle***, ***Rectangle*** și ***Square*** și conform diagramelor UML de mai jos (**2p**)
   b. În metoda statică main() a clasei Exercise3, se creează: (**2p**)
      - Câte o instanță a claselor ***Circle***, *Rectangle* și ***Square*** folosind constructorul 2. Se setează culoare cu clasa setter corespunzătoare. Se tipărește pentru fiecare obiect culoarea, aria și perimetrul acestuia



4. Scrieți un program Java cu blocuri ***try-catch*** care: (**1p**)
   - declară în ***try*** 2 variabile întregi cu valorile 10 și 0
   - tipărește rezultatul împărțirii
   - prinde în blocul ***catch*** excepția *ArithmeticException* și tipărește stiva de erori.

Temă pentru acasă:

5. Scrieți un program Java care implementează și testează (cu clasa Exercise5 cu metoda *main*) interfața și clasa definite de diagrama UML de mai jos, astfel: (**1p**)
   a. Se implementează interfața ***Movable*** și clasa ***MovablePoint*** conform diagramelor UML de mai jos (**0.5p**)
   b. În metoda statică main() a clasei Exercise5: (**0.5p**)
      - Se creează o instanță a clasei ***MovablePoint*** având punctul inițial la poziția (5, 5) și vitezele sunt (1, 1)
      - Se mutată punctul o dată în sus, apoi la dreapta, de 2 ori în jos și de 2 ori la stânga. Se tipărește obiectul ***MovablePoint***



6. Scrieți un program Java care: (**1p**)
   a. Implementează clasa ***TemperatureException*** având ca părinte clasa ***Exception,*** care are un singur constructor cu parametrul *message* de tip String (**0.5p**)
   b. În metoda statică main() a clasei Exercise3: (**0.5p**)
      - citește de la tastatură un număr întreg într-un bloc ***try-catch*** folosind clasa ***Scanner*** și ***Integer****.valueOf* (poate genera *NumberFormatException*)

```
Scanner s = new Scanner(System.in);
int t = Integer.valueOf(s.nextLine());
s.close();
```

- dacă temperatura este egală sau mai mare de 40 generează excepția ***TemperatureException*** cu mesajul *„Esti foarte bolnav!”*
- prinde în blocul ***catch*** excepția *NumberFormatException* și tipărește stiva de erori
- adaugă un bloc ***catch*** care prinde excepția *TemperatureException* și afișează mesajul de eroare.