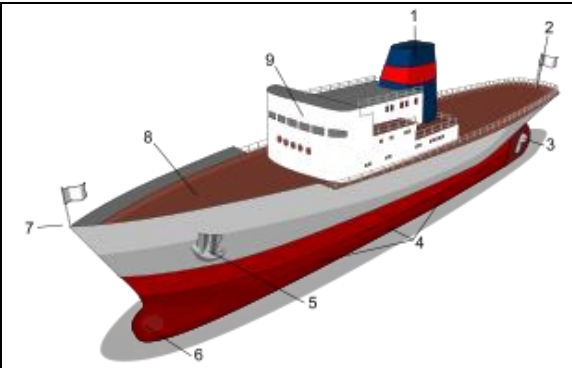





Laborator 3. Polimorfism

Nave maritime civile

Construiti o structura de clase corespunzatoare figurii urmatoare:

	
Nava	NavaCroaziera
nume, pavilion	nume, pavilion
	nrPasageri
	
Cargou	Feribot
nume, pavilion	nume, pavilion, nrPasageri
capacitateIncarcare	nrAuto

Fiecare nava specializata va fi reprezentata prin metodele

- `utilizare()` – se va afisa la ce serveste nava ('Croaziere de lux', 'Transport marfuri', 'Transport vehicule si pasageri',
- `toString()` – afiseaza toate informatiile depre nava respectiva

1. Scrieti clasa **Nava**, subclasa ei **NavaCroaziera** si o clasa **Flota** (4p)

Clasa Nava

Poate fi o clasa abstracta, care sa contina campurile `nume`, `pavilion`.

- Constructor: `Nava(nume, pavilion)`

- `toString()` – gasiti o solutie astfel incat sa poata fi apelata unitar din subclase
- `utilizare()` – metoda abstracta

Clasa NavaCroaziera

Extinde clasa `Nava`.

- o Constructor cu parametrii: nume, pavilion, nrPasageri
- o `utilizare()` – afiseaza 'Croaziere de lux'
- o `toString()` – returneaza un sir ca cel de mai jos:
`"NavaCroaziera - nume=Suceava, pavilion=RO, nrPasageri=1000"`

Se vor crea 2 nave de croaziera:

- Suceava, pavilion RO, maxim 1000 de pasageri
- Victoria, pavilion RO, 5000 de pasageri

Clasa Flota

Clasa Flota va contine un tablou de tip `Nava` cuprinzand toate navele din flota.

- o Constructor: `Flota()` – aloca un tablou `Nava nave[10]` (**obligatoriu in aceasta lucrare de laborator**) si nr. de nave egal cu 0
- o `toString()` – returneaza un sir cu componenta flotei
- o `utilizare()` – afiseaza utilizarea navelor din flota
- o `adaugaNava(Nava x)` – introduce nava x in flota (in tabelul `nave[]`)

Observatie. Daca la apelul metodei `adaugaNava()` se va constata ca nu mai este loc in tabelul `nave[]` atunci se va realoca acest tabel utilizand urmatoarea functie statica din clasa `Arrays`

`public static <T> T[] copyOf(T[] original, int newLength)`
 care realizeaza o copie a vechiului tabel completandu-l cu null daca noua lungime este mai mare.

Genericitate `<T>` este cunoscuta de la C++.

Metoda `main()` a clasei `Flota` poate fi urmatoarea:

```
public static void main(String[] args) {
    Flota flota1= new Flota();
    NavaCroaziera sv = new NavaCroaziera(1000, "Suceava", "RO");
    System.out.print(sv + "\nUtilizare:");
    sv.utilizare();
    System.out.println();

    flota1.adaugaNava( sv);
    flota1.adaugaNava(new NavaCroaziera(5000, "Victoria", "RO"));
    System.out.println(flota1);
    flota1.utilizare();
}
```

Iesirea produsa in acest caz va trebui sa fie astfel:

```
NavaCroaziera - nume=Suceava, pavilion=RO, nrPasageri=1000
Utilizare:Croaziere de lux

Flota:
```

```
1. NavaCroaziera - nume=Suceava, pavilion=RO, nrPasageri=1000
2. NavaCroaziera - nume=Victoria, pavilion=RO, nrPasageri=5000
```

Utilizare flota:

Suceava - Croaziere de lux

Victoria - Croaziere de lux

2. Scrieti clasele **Cargou** (extinde clasa **Nava**) si **Feribot** (extinde clasa **NavaCroaziera**) (3p pentru solutia optima)

Veti completa metoda **main()** din clasa **Flota** astfel incat sa se adauge in flota si o nava **Cargou** si una **Feribot**.

Flota:

```
1. NavaCroaziera - nume=Suceava, pavilion=RO, nrPasageri=1000
2. NavaCroaziera - nume=Victoria, pavilion=RO, nrPasageri=5000
3. Cargou - nume=Carpati, pavilion=RO, capacitateIncarcare=10000
4. Feribot - nume=Dunarea, pavilion=BG, nrPasageri=100, nrAuto=20
```

Utilizare flota:

Suceava - Croaziere de lux

Victoria - Croaziere de lux

Carpati - Transport marfuri

Dunarea - Transport vehicule si pasageri

3. Adaugati clasei **Flota** un nou constructor (2p)

Acesta va avea ca argument numele unui fisier din care se va prelua descrierea flotei. Pe prima linie va fi dat numarul de nave din fisier, iar pe celelalte linii descrierea navelor, cate una pe linie,

<nume_nava>, <pavilion>, <tip_nava>, ... numere intregi care caracterizeaza nava (tonaj, nr.pasageri, nrAuto, dupa tipul navei)

Veti folosi fisierul [flota.txt](#) care contine

```
11
Regina Maria, RO, NavaCroaziera, 1500
Carpati, RO, Cargo, 10000
Dunarea, BG, Feribot, 100, 20
Insula Misterioasa, RO, Feribot, 300, 75
Iasi, RO, NavaCroaziera, 2500
Apuseni, RO, Cargo, 20000
Plevna, BG, Cargo, 10020
Botosani, RO, NavaCroaziera, 500
Navodari, RO, Feribot, 300, 75
Constanta, RO, NavaCroaziera, 2500
Ultima limita, RO, Cargo, 50000
```

Veti presupune ca fisierul contine numai informatii corecte. Nu e nevoie sa le validati, ci doar sa verificati capacitatea tabloului nave[].

Introdceți în main() instrucțiunile:

```
Flota flota2 = new Flota("flota.txt");
System.out.println(flota2);
flota2.utilizare();
```

Lesirea produsă pentru acest fișier trebuie să fie următoarea:

```
Flota:
1. NavaCroaziera - nume=Regina Maria, pavilion=RO, nrPasageri=1500
2. Cargou - nume=Carpati, pavilion=RO, capacitateIncarcare=10000
3. Feribot - nume=Dunarea, pavilion=BG, nrPasageri=100, nrAuto=20
4. Feribot - nume=Insula Misterioasa, pavilion=RO, nrPasageri=300, nrAuto=75
5. NavaCroaziera - nume=Iasi, pavilion=RO, nrPasageri=2500
6. Cargou - nume=Apuseni, pavilion=RO, capacitateIncarcare=20000
7. Cargou - nume=Plevna, pavilion=BG, capacitateIncarcare=10020
8. NavaCroaziera - nume=Botosani, pavilion=RO, nrPasageri=500
9. Feribot - nume=Navodari, pavilion=RO, nrPasageri=300, nrAuto=75
10. NavaCroaziera - nume=Constanta, pavilion=RO, nrPasageri=2500
11. Cargou - nume=Ultima limita, pavilion=RO, capacitateIncarcare=50000

Utilizare flota:
Regina Maria - Croaziere de lux
Carpati - Transport marfuri
Dunarea - Transport vehicule si pasageri
Insula Misterioasa - Transport vehicule si pasageri
Iasi - Croaziere de lux
Apuseni - Transport marfuri
Plevna - Transport marfuri
Botosani - Croaziere de lux
Navodari - Transport vehicule si pasageri
Constanta - Croaziere de lux
Ultima limita - Transport marfuri
```

4. Adăugați clasei **Flota** o nouă metodă de conversie la sir (1p)

Această metodă va avea prototipul

```
public String toStringFlota(boolean dupaNume);
```

Dacă **dupaNume=true** atunci va rezulta un sir ca cel de mai jos (navele apar în ordinea lexicografică a numelui)

```
"Flota:
1. Cargou - nume=Apuseni, pavilion=RO, capacitateIncarcare=20000
2. NavaCroaziera - nume=Botosani, pavilion=RO, nrPasageri=500
3. Cargou - nume=Carpati, pavilion=RO, capacitateIncarcare=10000
4. NavaCroaziera - nume=Constanta, pavilion=RO, nrPasageri=2500
5. Feribot - nume=Dunarea, pavilion=BG, nrPasageri=100, nrAuto=20
6. NavaCroaziera - nume=Iasi, pavilion=RO, nrPasageri=2500
7. Feribot - nume=Insula Misterioasa, pavilion=RO, nrPasageri=300,
nrAuto=75
8. Feribot - nume=Navodari, pavilion=RO, nrPasageri=300, nrAuto=75
9. Cargou - nume=Plevna, pavilion=BG, capacitateIncarcare=10020
10. NavaCroaziera - nume=Regina Maria, pavilion=RO, nrPasageri=1500
11. Cargou - nume=Ultima limita, pavilion=RO, capacitateIncarcare=50000
"
```

Daca **dupaNume=false** atunci va rezulta un sir in care navele apar in ordinea lexicografica a pavilionului (intai cele cu BG si apoi cele cu RO).

Sugestii

1. Daca intampinati probleme la citirea fisierului va poate fi de folos [Chapter 6: File Processing](#) din cartea recomandata (se gaseste in sectiunea [Suplements](#))
2. In clasa **String** exista metodele trim(), **split()**
<https://docs.oracle.com/javase/9/docs/api/java/lang/String.html>
3. Utilizati clasele **StringBuilder**, **Arrays** si compararea obiectelor
<http://apollo.eed.usv.ro/~pentiuc/sd/comparareObiecte.pdf>
4. Privitor la sortarea/compararea obiectelor sugestii gasiti si in curs
<http://apollo.eed.usv.ro/~pentiuc/sd/comparareObiecte.pdf> (pag 7-15)