Obrada izuzetaka

- Problemi:
  - kod postaje nepregledan čim postoji nekoliko koraka koji mogu da rezultuju izuzetkom (duboko i nepregledno ugnežđivanje naredbi *if-then-else*)
  - funkcije koje vraćaju status ne mogu da vraćaju svoje "prirodne" rezultate, koji se zato moraju prenositi kao izlazni argumenti po referenci/preko pokazivača
  - pozivalac funkcija koje vraćaju kod greške može greškom ili namerno da ignoriše te greške: da ih ne proverava i ne obrađuje, ili da ih ne propagira svom pozivaocu
    - ispitivanje povratnih statusa, tj. postojanja grešaka, troši vreme i resurse najčešće bespotrebno
- Uzrok problema: obrada izuzetaka učešljana je u regularan tok, isprepletana je sa osnovnim tokom, pa ga čini nepreglednim i težim za programiranje
- Bolji pristup:
  - regularni tok programira se kao da je sve u redu, tj. da nema izuzetaka i nije opterećen njihovom obradom,
  - s tim da određeni koraci regularnog toka mogu da *bace* (*throw*) izuzetak
  - obrada izuzetaka se piše kao *obrađivač izuzetka* (*exception handler*) pridružen bloku sa regularnim tokom, koji *hvata* izuzetak (*catch*), ali nije izmešana sa njim
- Mehanizam obrade izuzetaka nije direktno vezan za objektno orijentisanu paradigmu, ali je podržan u svim današnjim popularnim OO programskim jezicima na vrlo sličan način, po istom principu

Obrada izuzetaka

```
enum DeviceException {deviceFaulty, communicationFailed};

Temperature readTemperature ();
Pressure readPressure ();
Humidiry readHumidity ();
…

void readMeteo () {

   try {

       Temperature temp = readTemperature();
       Pressure press = readPressure();
       Humidity hum = readHumidity();

       // Finally, do the job with temp, press, and hum

   }

   catch (DeviceException& de) {

     // Handle exceptions

   }

}
```