**Clase și Obiecte**

Pelle Remus-Nicolae

SDI Anul 1 Semestrul 2

**Obiectele** sunt cheia înțelegerii tehnologiei orientate pe obiecte. În jurul nostru, se află multe exemple de obiecte din lumea reală: câinele, biroul, televizorul, bicicleta.

Obiectele din lumea reală împărtășesc două caracteristici: toate au stare și comportament. Câinii au stare (nume, culoare, rasă) și comportament (lătrat, dat din coadă). Identificarea stării și a comportamentului pentru obiectele din lumea reală este o modalitate excelentă de a începe să gândiți în termeni de programare orientată pe obiecte.

Luați un minut și observați obiectele din lumea reală care se află în jurul tău. Pentru fiecare obiect pe care îl vedeți, puneți-vă două întrebări: „În ce stări posibile poate fi acest obiect?” și „Ce comportament posibil poate efectua acest obiect?”. Asigurați-vă că vă notați observațiile. Pe măsură ce faceți asta, veți observa că obiectele din lumea reală variază în complexitate; lampa de pe desktop poate avea doar două stări posibile (pornită și oprită) și două comportamente posibile (pornire, oprire), dar radioul de pe desktop poate avea stări suplimentare (pornit, oprit, volum curent, stație curentă) și comportament (pornire , oprire, măriți volumul, micșorați volumul, căutați, scanați și reglați). Puteți observa, de asemenea, că unele obiecte, la rândul lor, vor conține și alte obiecte. Aceste observații din lumea reală se traduc și în lumea programării orientate pe obiecte.

Obiectele digitale sunt similare din punct de vedere conceptual cu obiectele din lumea reală: sunt formate și ele din comportament de stare și comportament conex. Un obiect își stochează starea în câmpuri (variabile în unele limbaje de programare) și își expune comportamentul prin metode (funcții în unele limbaje de programare). Metodele funcționează pe starea internă a unui obiect și servesc drept mecanism principal pentru comunicarea obiect-obiect. Ascunderea stării interne și necesitatea efectuării tuturor interacțiunilor prin metodele unui obiect este cunoscută sub numele de încapsulare de date - un principiu fundamental al programării orientate pe obiecte.

Luați în considerare o bicicletă, de exemplu: atribuind starea (viteza actuală, cadența curentă a pedalei și treapta de viteză curentă) și oferind metode pentru schimbarea stării respective, obiectul rămâne sub controlul modului în care lumea exterioară are voie să o folosească. De exemplu, dacă bicicleta are doar 6 trepte, o metodă de schimbare a treptelor ar putea respinge orice valoare mai mică decât 1 sau mai mare decât 6.

Gruparea codului în obiecte digitale individuale oferă o serie de avantaje, inclusiv:

1. **Modularitate**: Codul sursă pentru un obiect poate fi scris și menținut independent de codul sursă pentru alte obiecte. Odată creat, un obiect poate fi trecut cu ușurință în interiorul sistemului.
2. **Ascunderea informațiilor**: interacționând numai cu metodele unui obiect, detaliile implementării sale interne rămân ascunse lumii exterioare.
3. **Reutilizarea codului**: dacă există deja un obiect (poate scris de un alt dezvoltator de software), puteți utiliza acel obiect în programul dvs. Acest lucru permite specialiștilor să implementeze / testeze / depaneze obiecte complexe, specifice sarcinilor, în care puteți avea încredere să ruleze în propriul cod.
4. **Conectare și ușurință de depanare**: dacă un anumit obiect se dovedește a fi problematic, îl puteți elimina pur și simplu din aplicație și conecta un alt obiect ca înlocuitor. Acest lucru este similar cu soluționarea problemelor mecanice în lumea reală. Dacă se rupe un șurub, înlocuiți șurubul, nu întreaga mașină.

În lumea reală, veți găsi adesea multe obiecte individuale, toate de același fel. Pot exista mii de alte biciclete, toate de aceeași marcă și model. Fiecare bicicletă a fost construită din același set de planuri și, prin urmare, conține aceleași componente. În termeni orientați spre obiecte, spunem că bicicleta dvs. este o instanță a clasei de obiecte cunoscute sub numele de biciclete. **O clasă** este planul din care sunt create obiecte individuale.

Proiectarea clasei Bicicletă se bazează pe discuția anterioară despre obiecte bicicletă. Câmpurile cadence, speed, și gear reprezintă starea obiectului, iar metodele (changeCadence, changeGear, speedUp etc.) definesc interacțiunea sa cu lumea exterioară.

Clasele sunt definite astfel:

|  |
| --- |
| class MyClass {  // field, constructor, and  // method declarations  } |

De asemenea, puteți adăuga modificatori, cum ar fi public sau private, chiar la început - astfel încât să puteți vedea că linia de deschidere a unei declarații de clasă poate deveni destul de complicată.

În general, declarațiile de clasă pot include aceste componente, în ordine:

1. **Modificatori** precum public, privat și o serie de altele pe care le veți întâlni mai târziu. (Cu toate acestea, rețineți că modificatorul privat poate fi aplicat doar claselor imbricate.)
2. **Numele clasei**, cu litera inițială cu majuscule prin convenție.
3. **extends** [numele clasei părinte] (superclasă), dacă există. O clasă (subclasa) poate extinde doar un părinte.
4. O listă separată prin virgule a interfețelor implementate de clasă, dacă există, precedată de cuvintele cheie **implements**. O clasă poate implementa mai multe interfețe.
5. Corpul clasei, înconjurat de acolade, {}.