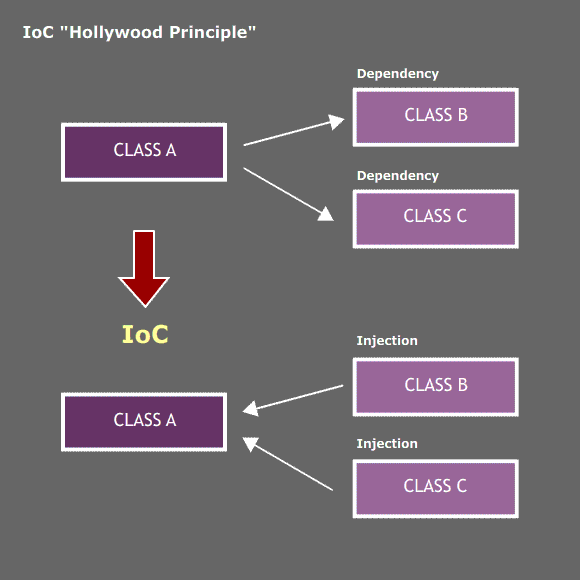
**TEMA 2 – PSSC**

**Inversion on Control Container**

În ingineria software-ului, **Inversarea Controlului** (IoC) este un principiu de proiectare în care porțiunile personalizate scrise ale unui program de calculator primesc fluxul de control dintr-un cadru generic. O arhitectură software cu acest design inversează controlul în comparație cu programarea tradițională procedurală: în programarea tradițională, codul personalizat care exprimă scopul programului solicită bibliotecile reutilizabile să aibă grijă de sarcini generice, dar cu inversarea controlului, care solicită codul personalizat sau specific de sarcină

Inversiunea controlului este folosită pentru a crește modularitatea programului și a face extensibilă și are aplicații în programarea orientată pe obiecte și alte paradigme de programare.

Termenul este legat de principiul de inversiune a dependenței, dar diferit de acesta, care se referă la decuplarea dependențelor dintre straturile de nivel înalt și nivelurile joase prin intermediul abstractizărilor partajate. Conceptul general se referă, de asemenea, la programarea bazată pe evenimente, prin faptul că este adesea implementată utilizând IoC, astfel încât codul personalizat se referă în general numai la gestionarea evenimentelor, în timp ce bucla evenimentului și expedierea evenimentelor / mesajelor sunt tratate de cadrul sau mediul de executare.



Inversiunea controlului este uneori numită în mod obișnuit "principiul de la Hollywood: Nu ne sunați, vă vom suna noi". :

De exemplu, cu programarea tradițională, funcția principală a unei aplicații ar putea face apeluri de funcții într-o bibliotecă de meniuri pentru a afișa o listă de comenzi disponibile și a solicita utilizatorului să selecteze una. Astfel, biblioteca va returna opțiunea aleasă ca valoare a apelului funcției, iar funcția principală utilizează această valoare pentru a executa comanda asociată. Acest stil era comun în interfețele bazate pe text. De exemplu, un client de e-mail poate afișa un ecran cu comenzi pentru a încărca e-mailuri noi, a răspunde e-mailului curent, a începe o nouă e-mail etc., iar execuția programului s-ar bloca până când utilizatorul apasă o tastă pentru a selecta o comandă.

Inversiunea controlului are următoarele scopuri de proiectare:

* Pentru a decupla executarea unei sarcini de la implementare.
* Pentru a focaliza un modul asupra sarcinii pentru care a fost proiectat.
* Pentru a elibera module de ipoteze despre modul în care alte sisteme fac ceea ce fac și în schimb se bazează pe contracte.
* Pentru a preveni efectele secundare la înlocuirea unui modul.

Cu inversarea controlului, pe de altă parte, programul va fi scris folosind un cadru software care cunoaște elemente comportamentale și grafice comune, cum ar fi sistemele de ferestre, meniurile, controlul mouse-ului și așa mai departe. Codul personalizat "umple golurile" pentru cadru, cum ar fi furnizarea unei tabele de elemente de meniu și înregistrarea unei subrutine de cod pentru fiecare element, dar este cadrul care monitorizează acțiunile utilizatorului și invocă subrutina atunci când este selectat un element de meniu . În exemplul clientului de poștă electronică, cadrul poate urmări ambele intrări de la tastatură și mouse și poate apela comanda invocată de utilizator prin oricare dintre mijloace și, în același timp, monitorizează interfața de rețea pentru a afla dacă sosesc mesaje noi și reîmprospătează ecranul atunci când activitatea de rețea este detectată. Același cadru ar putea fi folosit ca schelet pentru un program de calcul tabelar sau un editor de text. Dimpotrivă, cadrul nu cunoaște nimic despre browserele Web, foile de calcul sau editorii de text; implementarea funcționalității acestora necesită un cod personalizat.

În programarea orientată pe obiecte, există mai multe tehnici de bază pentru implementarea inversării controlului. Acestea sunt:

• Utilizarea unui model de localizare a serviciului

• Utilizarea injecției de dependență, de exemplu

• Injectarea constructorului

• Injectarea parametrilor

• Injecție de reglare

• Injectarea interfeței

• Utilizarea unei căutări contextualizate

• Utilizarea modelului de proiectare a metodei șablonului

• Utilizarea modelului de design strategic

**public** **class** **ServerFacade** {

**public** <K, V> V respondToRequest(K request) {

**if** (businessLayer.validateRequest(request)) {

DAO.getData(request);

**return** Aspect.convertData(request);

}

**return** **null**;

}

}