STRUKTURALNI PATERNI

1. ADAPTER PATERN

Adapter patern služi da se postojeći objekat prilagodi za korištenje na neki novi način u odnosu na postojeći rad, bez mijenjanja same definicije objekta. Na taj način obezbjeđuje se da će se objekti i dalje moći upotrebljavati na način kako su se dosad upotrebljavali, a u isto vrijeme će se omogućiti njihovo prilagođavanje novim uslovima.

- Primjer upotrebe Adapter paterna:

  Što se tiče Adapter paterna, još uvijek nismo uočili primjer za njegovu upotrebu u našem programu.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće:

  /

2. FACADE PATERN

Fasadni patern služi kako bi se klijentima pojednostavilo korištenje kompleksnih sistema. Klijenti vide samo fasadu, odnosno krajnji izgled objekta, dok je njegova unutrašnja struktura skrivena. Na ovaj način smanjuje se mogućnost pojavljivanja grešaka jer klijenti ne moraju dobro poznavati sistem kako bi ga mogli koristiti.

- Primjer upotrebe Facade paterna:

  Što se tiče Facade paterna, primjer za njegovu upotrebu smo uočili kod sastavljanja idealnog računara. Naime, na kupcu je samo da unese specifikacije željenog računara, a način na koji

se taj računar sastavlja je potpuno skriven od kupca koji vidi samo krajnji rezultat. Također i kod same opcije kupovine, kupac treba da vidi samo upozorenje o uspješnoj kupovini, a uklanjanje proizvoda iz korpe,regulisanje nove količine proizvoda, te eventualno uklanjanje proizvoda iako se dešavaju u pozadini, njihov način izvršavanja kupca se ne tiču, te trebaju biti skriveni od njega.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće:

1. Definisati novu klasu AkcijeFasada koja će implementirati metode sastaviRačunar() i kupiProizvode();

2. Promijeniti implementaciju kupca na način da se samo pozivaju gotove metode novodefinisane klase;

3. DECORATOR PATERN

Decorator patern služi za omogućavanja različitih nadogradnji objektima koji svi u osnovi predstavljaju jednu vrstu object (odnosno, koji imaju istu osnovu). Umjesto da se definiše veliki broj izvedenih klasa, dovoljno je omogućiti različito dekoriranje objekata (tj. Dodavanje različitih detalja), te se na taj način pojednostavljuje I rukovanje objektima klijentima, i samo implementiranje modela objekata.

- Primjer upotrebe Decorator paterna:

Što se tiče Decorator paterna, primjer za njegovu upotrebu još uvijek nismo uočili u našem programu.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće:

/

4. BRIDGE PATERN

Bridge patern služi kako bi se apstrakcija nekog objekta odvojila od njegove implementacije. Ovaj patern veoma je važan jer omogućava ispunjavanje Open-Closed SOLID principa, odnosno uz poštivanje ovog paterna omogućava se nadogradnja modela klasa u budućnosti te osigurava da se neće morati vršiti određene promjene u postojećim klasama.

- Primjer upotrebe Bridge paterna:

  Što se tiče Bridge paterna, mogući primjer za njegovu upotrebu smo uočili kod same cijene proizvoda koji se nalaze u korpi ili koje je kupac odlučio da kupi. Naime, kupac u našem ITShopu može biti Student pri čemu se ukupna cijena računa uzimajući u obzir popust, ali uvijek sa istom osnovicom (navedenom cijenom proizvoda). To nam ne bi stvaralo problem ukoliko bi željeli uvesti popuste i sniženja za još neke skupine kupaca (npr. kupcima koji su već određeni period registrovani na naš ITShop), međutim ukoliko bismo se odlučili u budućnosti dodati mogućnost raznih saradnji, što bi dovodilo do različitih dogovora oko određenih cijena (što je danas jako popularno kod online shoppinga) uvođenje ovog paterna bi nam znatno pomoglo pri prikazu cijene na ispravan način, jer bi to zahtijevalo samo novu metodu (novu apstrakciju), na koju će se naslanjati konkretna implementacija izračuna cijene za takve kupce.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće:

  1. Dodati novi interfejs ICijena koji će sadržavati definiciju metode za izračun cijene određenog proizvoda;

2. Dodati novu klasu Bridge, koja će sadržavati apstrakciju i kojoj će jedino kupac imati pristup;

5. COMPOSITE PATERN

Composite patern služi za kreiranje hijerarhije objekata. Koristi se kada svi objekti imaju različite implementacije nekih metoda, no potrebno im je svima pristupati na isti način, te se na taj način pojednostavljuje njihova implementacija.

- Primjer upotrebe Composite paterna:

  Što se tiče Composite paterna, primjer za njegovu upotrebu smo uočili kod samih proizvoda. Posmatrajući naš dijagram klasa, uvidjeli smo kod klasa proizvoda da ih jako dosta koristi istu metodu getTip(). Bez obzira na različite ishode napisane metode, svakoj je potrebno pristupati na isti način, te uvođenjem Composite paterna smatramo da bi se znatno pojednostavio I smanjio program.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće:

  1. Definisati interfejs ITip koji će sadržavati definiciju metode za dobivanje tipa proizvoda;

  2. Naslijediti ga od strane svih klasa proizvoda koje su sadržavale prethodno napisanu metodu getTip();

6. PROXY PATERN

Proxy patern služi za dodatno osiguravanje objekata od pogrešne ili zlonamjerne upotrebe. Primjenom ovog paterna omogućava se kontrola pristupa objektima, te se onemogućava manipulacija

objektima ukoliko neki uslov nije ispunjen, odnosno ukoliko korisnik nema prava pristupa traženom objektu.

- Primjer upotrebe Proxy paterna:

  Što se tiče Proxy paterna, primjer za njegovu upotrebu smo uočili kod korisnika.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće:

7. FLYWEIGHT PATERN

Flyweight patern koristi se kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji svi u suštini predstavljaju jedan objekat. Samo ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama (tzv. specifično stanje), vrši se njegova instantacija, dok se u svim ostalim slučajevima koristi postojeća opća instance objekta (tzv. bezlično stanje). Korištenje ovog paterna veoma je korisno u slučajevima kada je potrebno vršiti uštedu memorije.

- Primjer upotrebe Flyweight paterna:

  Što se tiče Flyweight paterna, primjer za njegovu upotrebu smo uočili kod upravljanja proizvodima.

- Kako bi se to postiglo, potrebno je izvršiti sljedeće: