Uzorci dizajna - 2. Zadaća

Chain of responsibility

Ovaj uzorak je odabran za dvije svrhe. Prva je odabrati jedan od tri algoritma za generiranje urona, a svaki od tih algoritama ima mogućnost izvršiti zahtjev - pozvati evidentiranje generiranih urona i ispis u datoteku. (Tekst iz zadaće: "...tijekom kojeg će se odabrati jedan od tri ponuđena algoritma za sastavljanje popisa"). Druga svrha je odabrati koju će se federaciju obavijestiti o uronu pojedinog ronioca koji pripada toj federaciji. (Tekst iz zadaće: "Treba pretpostaviti da nisu unaprijed poznate sve agencije/federacije kao ni druge institucije koje treba obavijestiti o uronu ronioca.").

Ovaj uzorak dozvoljava da se proslijedi zahtjev lancu objekata i da jedan od njih prihvati i obradi traženi zahtjev, a to je upravo ono što je ovdje implementirano. "AlgorithmHandler" je jedan od objekata u lancu za odabir algoritama i implementira sučelje "IHandler", dok je "FederationObserver" jedan od objekata u lancu za obavijesti, a također implementira sučelje "IHandler". "ChainGenerator" je klasa koja je zadužena postavljanje lanca za odabir algoritma, dok je sam "DivingClub" zadužen za istu stvar kod generiranja lanca za obavijesti.

Observer

(Tekst iz zadaće: "Ronilački klub obavezan je nakon svakog urona obavještavati agenciju/federaciju kojoj pripada ronioc".) Observer uzorak omogućava definiranje 1:N veze pri čemu subjekt nakon svake promjene određenog stanja obavještava svoje "pretplatitelje" (observere) o promjeni stanja, a to je upravo što se ovdje traži. "DivingClub" (ronilački klub) implementira sučelje "Subject" i nakon što neki ronioc pristupi uronu, obavještava potrebne federacije i agenciju, u ovom slučaju objekte klase "FederationObserver" koji implementiraju sučelje "Observer".

Visitor

Omogućuje dodatnu operaciju nad postojećim klasama, a ono što je ovdje implementirano po mom izboru je ispis broja ronioca koji su sudjelovali u uronima i njihovu prosječnu dubinu urona prema određenoj federaciji. Sučelje “DiverElement” definira operaciju za prihvat određenog visitora, a implementira je klasa “Diver”. “DiverVisitor” je sučelje za “posjetu” određenog ronioca, odnosno pristup njegovim podacima i metodama, a implementira je “DiverConcreteVisitor” koji računa broj ronioca i prosječnu dubinu.

Promjene u odnosu na prvu zadaću

Prva promjena u odnosu na prvu zadaću je uvođenje Factory methoda za generiranje objekata za pojedini algoritam, a za to je zadužen “WorkerFactory”. Druga promjena je detaljniji i smisleniji Builder za kreiranje ronioca (unutarnja klasa “DiverBuilder”).

Specifičnosti vezane za implementaciju

Odabir algoritma (Chain of responsibility): svaki algoritam prvo računa svoju ukupnu razinu sigurnosti te dobiva određeni ID koji je u stvari razina sigurnosti, a ronilački klub ujedno zapisuje taj ID prema kojem se u lancu algoritama uspoređuje odgovara li ID algoritma zapisanom te ako odgovara obrađuje zahtjev.

Observer: ronilački klub je ujedno subjekt (topic), na koji se “pretplaćuju” observeri (federacije). Ronilački klub vodi evidenciju o roniocima koji su pristupili određenoj federaciji (implementirano kao mapa) i obavještava potrebne federacije (uvijek se obavještava HRS, a osim njega pripadajuća federacija - chain of responsibility). Kada uronu pristupi određeni ronioc za kojeg još ne postoji federacija, ona se kreira i dodaje u lanac odgovornosti.

Dijagram rješenja