**Poltergeist Class**

**Definícia**

Názvom Poltergeist (nazývaná aj Gypsy class) sa označuje nadbytočná trieda v kóde, ktorá pridáva komplexitu a zložitosť programu a zaberá zdroje pre vývoj, beh, údržbu a testovanie programu.

**Charakteristika**

Charakteristické vlastnosti Poltergeist tried sú ich krátka (obmedzená) životnosť, ktorá vyplýva z faktu, že tieto triedy sú špecifické na a použiteľné len v istej časti kódu. Vo väčšine prípadov tieto triedy nemajú vlastné „responsibilities“ (tieto triedy samy osebe neovplyvňujú beh aplikácie a premenných), sú bezstavové a iba prepájajú a inicializujú metódy a funkcie z iných, dlhodobejších tried. Keďže sú tieto triedy špecifické na časť programu, ich prítomnosť môže mať neželané vedľajšie účinky počas behu.

**Príčina vzniku a identifikácia**

Za vytvorením takejto triedy môže byť zlý návrh programu, kedy programátor očakáva oveľa väčšiu zložitosť softvéru, využívanie predlohy „templatu“ pre iný, komplexnejší softvér alebo aj nedostatok skúseností programátora s objektovo orientovaným programovaním.

Tieto triedy môžeme často identifikovať podľa ich názvu, ktorý v sebe obyčajne zahŕňa slová ako „controller“, „handler“ a iné. Problémom pri identifikácií (a pomôckou pri hľadaní) je obvykle aj zložitosť tejto triedy, kedy programátor nevie alebo má problém určiť zmysel a funkcionalitu tejto triedy a jej celkový význam v rámci programu.

Veľmi zjednodušeným príkladom pre Poltergeist triedu by mohol byť výpočet štvorca:



V tomto príklade by namiesto celej triedy stačila jednoduchá funkcia:



Iným príkladom by mohol byť jednoduchý systém na vytvorenie a správu užívateľov softvéru.

Poltergeist trieda, ktorú sledujeme sa nazýva UserController a UserService:





Preskúmaním tejto triedy je zjavné, že sa nedá tak jednoducho určiť, akú úlohu v rámci programu plní a aký je jej prínos. Je vidieť, že táto trieda využíva metódy inej triedy (UserService) v metóde createUser alebo findAll.

**Riešenie**

Metóda zbavovania sa Poltergeist tried sa nazýva Ghostbusting. V princípe je dôležité refraktorovanie kódu a pri narazení na takúto triedu je potrebné zbaviť sa jej a funkcionalitu tejto triedy premiestniť do tried, ktorú táto trieda volala a využívala.

Pre ukážkový príklad to znamená, odstránenie triedy UserController a refraktorovanie triedy UserService tak, aby sa nestratila funkcionalita pridaná Poltergeist triedou, napríklad pre metódu findAllUsers():



Prevencia

Je dôležité rozlišovať či by pridanie novej triedy do programu zjednodušilo dizajn pridalo mu hodnotu alebo iba zvýšilo komplexnosť bez pridania benefitov.

Zdroje

Rectangle príklad:

<https://gist.githubusercontent.com/besbes-ahmed/939cde532cd3b17a6949bd6cfc72ed17/raw/0a2c3c49f980f26c68039dae902efe2bb88c07e5/example_18.py>

UserClass príklad:

<https://codegym.cc/groups/posts/320-what-are-anti-patterns-lets-look-at-some-examples-part-1>