1. Concert //mostenire Eveniment
2. Manager

**Coeziune:** Ignorand nerespectarea SRP, clasa face ce ar trebui sa faca; avem o coeziune mica.

**Cuplaj:** Clasa are un cuplaj mare, avand rol de intermediar pentru majoritatea metodelor celorlaltor clase.

**SRP:** Principiul nu se repecta. Manager are atat rol de gestionare a biletelor, cat si de gestionare a conturilor. Solutia ar fi splitarea in 2 clase, TIcketManager si AccountManager, asignarea metodelor fiind subinteleasa.

**OCP:** Nu este cazul. Divizarea clasei nu ar presupune existenta unei interfete, considerand faptul ca metodele sunt disjuncte.

**LSP:** Analog ORP. Nu e necesare o extensie.

**ISP:** Se respecta principial. Dar splitarea ar duce la o acuratete mai mare a acestui principiu.

**DIP:** Avem constructor implicit.

1. Cont

**Coeziune:** Clasa abstracta, extinsa de User si Guest. Prezinta setteri, getteri si o functie de buyticket folosita de mostenitori. Coeziune mare.

**Cuplaj:** Are 2 mostenitori si este agregata de Manager. Cuplaj mare.

**SRP:** Principiul se respecta. Ca si clasa abstracta, implementeaza metodele folosite de mostenitorii sai.

**OCP:** O singura exceptie, retusabila : ambii mostenitori folosesc o metoda requireticket ce ar putea fi implementata in cadrul clasei abstracte cont. In rest, clasa nu trebuie modificata, ea fiind extinsa de cei doi mostenitori.

**LSP:** E o clasa abstracta reusita, fiecare dintre mostenitori fiind refactorizabili la aceasta si neavand clauze conditionale de verificare a tipurilor acestora. Asadar, principiul este respectat.

**ISP:** Nu avem metode nefolosite de mostenitori. Principiul este respectat.

**DIP:** Nu depinde de mostenitorii sai, ci acestia de ea. Principiul este respectat.

1. Eveniment

**Coeziune:** Metodele implementate respecta cerintele legate de functionalitatile acestei clase. Coeziune mare.

**Cuplaj:** Are o clasa extensie si este agregata de Manager. Cuplaj mic.

**SRP:** Corespunzator descrierii alocate la coeziune, clasa isi respecta limitele functionalitatii impuse. Se respecta principiu.

**OCP:** Clasa extensie implementeaza un numar mai mare de metode. Considerand totusi ca am putea extinde Eveniment si la alte clase distincte, nu este neaparat eronat in context mai dezvoltat. Insa contextual, metodele extensiei puteau fi implementate si in clasa noastra.

**LSP:** Considerand ca extensia este unica, principiul se respecta : Concert este refactorizabil la Eveniment.

**ISP:** Nu avem metode nefolosite de mostenitori.

**DIP:** Mostenitorul depinde de clasa Eveniment.

1. Guest //mostenire Cont
2. User//mostenire Cont