**IZVJEŠTAJ – PRINCIPI DOBROG DIZAJNA**

1. **Princip pojedinačne odgovornosti** (engl. Single Responsibility Principle-SRP) glasi:  
   Klasa bi trebala imati samo jedan razlog za promjenu.

Dakle, nijedna klasa ne bi trebala da zna previše informacija jer na taj način dolazi do krhkosti programa. Treba raspodijeliti poslove među klasama, što je i urađeno. Npr. klasa *Kokice* je apstraktna i nju nasljeđuju *SlatkeKokice* i *SlaneKokice*, jer nije potrebno da se klasa *Kokice* bavi cijenama različitih vrsta kokica. Ovim smo postigli podjelu odgovornosti o podacima.

1. **Otvoreno zatvoren princip** (engl. Open Closed Principle – OCP ) glasi:  
   Entiteti softvera (klase, moduli, funkcije) trebali bi biti otvoreni za nadogradnju, ali zatvoreni za modifikacije.

To znači da trebamo biti sposobni mijenjati okruženje oko modula, bez promjene samog modula.

Ovaj prinip je ispoštovan korištenjem interfejsa *InotifyPropertyChanged* umjesto da smo koristili neku klasu za istu namjenu.

1. **Liskov princip zamjene** (engl. Liskov Substitution Principle – LSP) glasi:  
   Podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima. Apstraktna klasa treba da se može koristiti u svim situacijama kao i izvedena klasa. Ukoliko postoji makar i jedna apstraktna metoda koja se ne može primijeniti na izvedenu klasu, Liskov princip nije zadovoljen i klasa koja je izvedena zapravo ne bi trebala biti izvedena.  
   Sve izvedene klase se mogu koristiti u svim situacijama kao i apstraktna klasa, i obrnuto. Dakle, LSP je ispoštovan.
2. **Princip inverzije ovisnosti** (engl. Dependency Inversion Principle – DIP) glasi:   
   **A.** Moduli visokog nivoa ne bi trebali ovisiti od modula niskog nivoa, oba bi trebalo da ovise od apstrakcija.   
   **B.** Moduli ne bi trebali ovisiti od detalja. Detalji bi trebali ovisiti od apstrakcija.

Apstraktne klase i interfejsi se mnogo manje mijenjaju nego njihove konkretne izvedenice, pa je bolje ovisiti o apstrakcijama nego o klasama.  
Npr. umjesto da smo imali zasebne klase *SlatkeKokice* i *SlaneKokice*, napravljena je apstraktna klasa *Kokice* i njene konkretne izvedene klase *SlatkeKokice* i *SlaneKokice,* tako da će sve promjene se dešavati u klasi *Kokice*.

1. **Princip izoliranja interfejsa** (engl. Interface Segregation Principle – ISP) glasi:  
   Klijenti ne treba da ovise o metodama koje neće upotrebljavati.  
   Npr. klasa *Karta* je „debela“ i klijentu bi bilo zamorno pratiti sve njene promjene. Zato je klijentu omogućeno da koristi interfejsi i vidi samo metode koje se tiču njega direktno.