**Dependency Inversion Prensibi**

Dependency Inversion (Bağımlılığı Ters Çevirme) prensibi sınıflar arası bağımlılığın olabildiğince az olmasını ve üst seviye sınıfların alt seviye sınıflara bağımlı olmaması gerektiğini önermektedir. Yüksek seviyeli bir sınıfta gereksinimden doğan değişimler alt seviye sınıfları bunu uymaya zorlamaktadır. Ancak alt seviyeli bir sınıftaki değişimler yüksek seviyeli bir sınıfı hiçbir şekilde etkilememesi gerekmektedir.

Sınıflar arası bağımlılığın azaltılması için uygulamanın prensiplere uygun olarak tasarlanmasını gerektirir. Eğer bağımlılık olacak ise bu bağımlılık alt seviye sınıfların üst seviye sınıflara bağımlılığı şeklinde olmalıdır.

**Uygulama**

public class Kullanici {  
  
 private String mail;  
 private String ip;  
 private String telefon;  
 private String isim;  
  
 public Kullanici() { }  
 public Kullanici(String isim, String mail, String telefon, String ip) {  
 setMail(mail);  
 setTelefon(telefon);  
 setIsim(isim);  
 setIp(ip);  
 }  
  
 public String getIp() {  
 return ip;  
 }  
  
 public void setIp(String ip) {  
 this.ip = ip;  
 }  
  
 public String getMail() {  
 return mail;  
 }  
  
 public void setMail(String mail) {  
 this.mail = mail;  
 }  
  
 public String getTelefon() {  
 return telefon;  
 }  
  
 public void setTelefon(String telefon) {  
 this.telefon = telefon;  
 }  
  
 public String getIsim() {  
 return isim;  
 }  
  
 public void setIsim(String isim) {  
 this.isim = isim;  
 }  
}

public interface Iletisim {  
  
 void bilgiGonder(Kullanici kullanici, String mesaj);  
}

public class Haber {  
  
 private Mail mail;  
 private Bildirim bildirim;  
 private ArrayList<Iletisim> iletisimListesi;  
  
 public Haber() {  
 iletisimListesi = new ArrayList<>();  
 mail = new Mail();  
 bildirim = new Bildirim();  
 iletisimListesi.add(mail);  
 iletisimListesi.add(bildirim);  
 }  
  
 public void bilgilendir(Kullanici kullanici, String mesaj) {  
  
 for(Iletisim iletisim : iletisimListesi) {  
 iletisim.bilgiGonder(kullanici, mesaj);  
 }  
 */\*  
 mail.mailGonder(kullanici.getMail(), mesaj);  
 bildirim.bildirimGoster(kullanici.getIp(), mesaj);  
 \*/* }  
}

public class Mail implements Iletisim {  
  
 public void mailGonder(String mail, String mesaj) {  
 System.*out*.println(mail + " adresine " + mesaj + " gönderildi.");  
 }  
  
 @Override  
 public void bilgiGonder(Kullanici kullanici, String mesaj) {  
 mailGonder(kullanici.getMail(), mesaj);  
 }  
}

public class Bildirim implements Iletisim {  
  
 public void bildirimGoster(String ip, String mesaj) {  
 System.*out*.println(ip + " adresine " + mesaj + " gönderildi.");  
 }  
  
 @Override  
 public void bilgiGonder(Kullanici kullanici, String mesaj) {  
 bildirimGoster(kullanici.getIp(), mesaj);  
 }  
}

public class Main {  
  
 public static void main(String[] args) {  
  
 Kullanici kullanici = new Kullanici("Sefa Aras", "sefaaras@ktu.edu.tr", "0", "192.1.1.1");  
 Haber haber = new Haber();  
  
 haber.bilgilendir(kullanici, "Deneme Mesaj");  
  
 }  
}