2023-2024 Bahar
Nesneye Dayalı Programlama I
Bütünleme Sınavında Çıkabilecek Örnek Sorular ve Cevapları

Örnek Soru 1:

Bir Üniversite sınıfı yazın. (Türkçe karaktersiz olarak; `Universite`) Kodunuzun başında dosya adını belirtin.
`Universite` sınıfınızın;

`String` yani karakter katarı tipinde `private` bir `ad`,
`String` yani karakter katarı tipinde `private` bir `bulunduguSehir`,
`int` yani tamsayı tipinde `private` bir `kurulusYili`,

şeklinde alanları(field) olsun. `Universite` sınıfınız, bu alanlar için getter ve setter metotlara sahip olsun.

Örnek Soru 1'in Cevabı:

Universite.java

```
public class Universite{

	private String ad;
	private String bulunduguSehir;
	private int kurulusYili; //Önceki versiyonda tipi yanlışlıkla
String yazılmış


	public String getAd() {
		return ad;
	}
	public void setAd(String ad) {
		this.ad = ad;
	}
	public String getBulunduguSehir() {
		return bulunduguSehir;
	}
	public void setBulunduguSehir(String bulunduguSehir) {
		this.bulunduguSehir = bulunduguSehir;
	}
	public int getKurulusYili() { //Dolayısıyla bu metodun dönüş
tipi int oldu
		return kurulusYili;
	}
	public void setKurulusYili( int kurulusYili) { //Bu metodun
parametresinin tipi int oldu.
		this.kurulusYili = kurulusYili;
```

```
    }
}
```

Örnek Soru 1'in Cevabı için Açıklamalar:

YÖK gibi birden fazla Üniversite verisiyle ilgilenmek durumunda olan bir kurumda yazılım geliştirici olduğumuzu düşünelim. Türkiye'deki bugün için 204 adet üniversitenin verilerini bir nesne yönelimli programlama dili olan Java kullanarak saklamak, işlemek, güncellemek gereksinimiyle, `Universite` adlı bir sınıfımız olması gerekmektedir. Universitelerin adlarını `Universite` sınıfımızın `ad` değişkeninde tutmak iyi bir fikir olarak görünmektedir. Böylece sınıfımızdan iş süreçleri içindeki program akışlarımızda nesne türettiğimizde, üniversitelerin adlarını saklayacağımız değişkenlerimiz hazırlanmış olur.

Yukarıdaki kodun içinde

```
private String ad;
```

satırını bulunuz.

Bu satırda `ad` isimli bir alan tanımlanmaktadır. Bunu yaparken yazılımınızda yeni `Universite` nesneleri türeteceğinizi ve bunların `ad` şeklinde, sisteme eklemek isteyeceğimiz üniversitenin adını karakter katarı(`String`) olarak tutacak bir alan olacağını varsaymaktayız. `private` tanımı ilgimizi çekmesi gereken bir diğer belirtimdir.

Yukarıdaki kodun içinde `setAd` ifadesini bulun. Bu ifadenin yer aldığı

```
    public void setAd(String ad) {
        this.ad = ad;
    }
```

öbeğine `ad` alanının ya da fieldının setter'ı denir. Bu bir metottur. Bu setter metotlarına `ad` alanını `private` olarak tanımladığımız için ihtiyaç duyulur.

Örneğin ana program akışınızda;

bir `Universite` nesnesi türettiğinizi düşünün:

```
Universite uniArel = new Universite();
uniArel.ad = "Arel Üniversitesi";
```

`ad` alanına bu şekilde ulaşmanızı engelleyecek olan kavram tanımın yapıldığı satırdaki `private`'tır. private yüzünden yukarıdaki şekilde ad alanına ulaşılmaz. Bunun yerine, nesnemize adını ekleme işlemini program akışımızda `setAd` metodu kullanarak şu şekilde yaparız:

```
    Universite uniArel = new Universite();
    uniArel.setAd("Arel Üniversitesi");
```

Bunu neden böyle yaptığımıza bir örnek olarak, bunun bir protokol olduğu söylenebilir. Nesne yonelimli programlama dünyasındaki bu protokole gardrobunuzdaki bir giysiye ulaşma örneği üzerinden bakılabilir. Gardrobunuza bir giysi eklemek için gardrobunuzun tavanını kırmazsınız; kapısını açarsınız ve giysinizi eklersiniz.

Benzer şekilde;

```
public String getAd() {
     return ad;
}
```

metodu bir getter olarak adlandırılır. Ve bu da bir metottur. Bir `Universite` nesneniz olduğunu ve bunun adına ihtiyaç duyduğunuzu düşünelim.

```
System.out.println(uniArel.getAd());
```

şeklinde `uniArel` isimli nesnenizin `ad` alanına ulaşıp yazdırabilirisiniz.

Üniversitelerin bulunduğu şehir bilgisini saklamak için de `bulunduguSehir` alanı benzer şekilde kullanılır.

Örnek Soru 2:

Bir Soru sınıfı yazın. Sınıfınızın kodlarına başlamadan önce sınıf dosyanızın adını belirtin.
`Soru` sınıfınızın;

`String` tipinde `soruMetni` alanı(fieldı),
`String` tipinde `cevapMetni` alanı(fieldı),
`int` tipinde `yuzdeEtkisi` alanı(fieldı)

bulunsun. Bu alanlar `private` olsun. Sınıfın bütün alanlar için getter ve setter metotları bulunsun.

Örnek Soru 2'nin Cevabı:

Soru.java

```
public class Soru {
     private String soruMetni;
     private String cevapMetni;
     private int yuzdeEtkisi;

     public String getSoruMetni() {
          return soruMetni;
     }
     public void setSoruMetni(String soruMetni) {
```

```
            this.soruMetni = soruMetni;
      }
      public String getCevapMetni() {
            return cevapMetni;
      }
      public void setCevapMetni(String cevapMetni) {
            this.cevapMetni = cevapMetni;
      }
      public int getYuzdeEtkisi() {
            return yuzdeEtkisi;
      }
      public void setYuzdeEtkisi(int yuzdeEtkisi) {
            this.yuzdeEtkisi = yuzdeEtkisi;
      }

}
```

Örnek Soru 3:

Tarayıcı (Chromium, Firefox, vb.) isimli bir sınıfın kodunu yazınız. Dosya adını kodun en başına yazınız.
Sınıfın;

`String` tipinde `ad`,
`String` tipinde `gelistirenKurulus`,

şeklinde `private` alanları olsun.

Örnek Soru 3'ün cevabı:

Tarayici.java

```
public class Tarayici {
      private String ad;
      private String gelistirenKurulus;

      public String getAd() {
            return ad;
      }
      public void setAd(String ad) {
            this.ad = ad;
      }
      public String getGelistirenKurulus() {
            return gelistirenKurulus;
      }
      public void setGelistirenKurulus(String gelistirenKurulus) {
            this.gelistirenKurulus = gelistirenKurulus;
      }
}
```

Örnek Soru 4:

İki tamsayı (`int`) değer parametre alan ve bu değerlerin farkını hesaplayan bir metodu bir Java sınıfı içinde yazın. Java sınıfınızın `main` metodu içinde kullanıcıdan iki tam sayı değer istensin, önceden yazılan çıkarma işlemi yapan metodu `main` metodu içinde dışarıdan girilen parametrelerle çağırın. Kodun başına dosyanızın adını yazınız.

Örnek Soru 4'ün Cevabı:

CikarmaMetodu.java

```
import java.util.*;


public class CikarmaMetodu{

     public static int cikar(int sayi1, int sayi2){
          return sayi1 - sayi2;
     }

     public static void main(String[] args){
          Scanner sc = new Scanner(System.in);
          System.out.println("Eksilen sayıyı giriniz:");
          int sayi1 = sc.nextInt();
          System.out.println("Çıkan sayıyı giriniz:");
          int sayi2 = sc.nextInt();
          int sonuc = cikar(sayi1,sayi2);
          System.out.println("Çıkarma İşlemi Sonucu:" + sonuc);

     }

}
```

Örnek Soru 5:

Bir kısmı aşağıdaki kod parçasında verilen bir `Calisan` sınıfınız bulunuyor. Bütün alanları ilklendiren bir kurucu(constructor) metodunu yazınız. Ayrıca `toString` metodunu yazınız.

Calisan.java

```
public class Calisan {

    private String ad;
    private String soyad;
    private int maas;

 …
```

Örnek Soru 5'in Cevabı:

Kurucu:

```
public Calisan(String ad, String soyad, int maas) {
     this.ad = ad;
     this.soyad = soyad;
     this.maas = maas;
}
```

toString:

```
@Override
public String toString(){
    String res = "";
    res = "Calisan adi, soyadi:" + this.ad + " " + this.soyad + "\
n";
    res += "Calisan maasi:" + this.maas + "\n";
    return res;
}
```

Örnek Soru 6:

Bir kısmı aşağıdaki kod parçasında verilen bir `Universite` sınıfınız bulunuyor. Bütün alanları ilklendiren bir kurucu(constructor) metodunu yazınız. Ayrıca `toString` metodunu yazınız.

Universite.java

```
public class Universite {

     private String ad;
     private int kurulusYili;
     private int bulunduguSehirKodu;

 …
}
```

Örnek Soru 6 Cevabı:

Kurucu:

```
public Universite(String ad, int kurulusYili, int
bulunduguSehirKodu) {
          super();
          this.ad = ad;
          this.kurulusYili = kurulusYili;
          this.bulunduguSehirKodu = bulunduguSehirKodu;
     }
```

toString:

```
@Override
     public String toString() {
          return "Universite [ad=" + ad + ", kurulusYili=" +
kurulusYili + ", bulunduguSehirKodu=" + bulunduguSehirKodu
                    + "]";
     }
```

Örnek Soru 6 Cevabının Açıklaması

Kurucu ya da İngilizcesi *constructor* olan bir metot çeşidi vardır.
Diğer sorulardaki toplama yapan metot, çıkarma yapan metot gibidir. Ancak onlardan farklı olarak bir dönüş tipi yoktur.

Kurucu denilen metot bu sınıftan bir tane yapmamızı sağlar. Örneğin programımızın başka bir yerinde şöyle bir satır kullanmak istediğimizi düşünelim;

```
Universite bizimUniversitemiz = new Universite();
```

Bu satırı yazabilmemizi sağlayan bu parametresiz kurucudur. Parametresiz bir kurucu şu şekildedir:

```
public Universite(){

}
```

Yolun başında 2007 yılındayken henüz sadece Üniversite'nin adı varken şu şekilde bir yeni `Universite` nesnesi hazırlamak istediğimizi düşünelim:

```
Universite benimUniversitem = new Universite("Arel Üniversitesi");
```

Bunun için sadece ad alanını ilklendiren, ad alanına değer atayacak olan bir kurucuya ihtiyaç duyulur:

```
public Universite(String ad) {
          super();
```

```
        this.ad = ad;
    }
```

Ya da daha 2000'li yılların başında, henüz üniversitenin adı bile yokken 2007 yılında bir üniversite kurma tasarısının dijitalleştirildiğini düşünelim. Bu tasarı için sadece kuruluş yılını belirleyen bir kurucuya ihtiyaç duyulur:

```
    public Universite(int kurulusYili) {
        super();
        this.kurulusYili = kurulusYili;
    }
```

Her üç alanın değerlerinin belirli olduğu, yani üniversite adının, kuruluş yılının ve bulunacağı şehrin belli olduğu bir dönemde üniversite nesnemizi dijitalleştirmeye karar verilmiş olunabilir.

```
Universite benimUniversitem = new Universite("Arel Üniversitesi", 2007, 34);
```

Böyle bir satırı programımızda yazabilmek için, Univeriste sınıfı içinde aşağıdaki kurucu metodun yazılması gerekir:

```
public Universite(String ad, int kurulusYili, int bulunduguSehirKodu) {
        super();
        this.ad = ad;
        this.kurulusYili = kurulusYili;
        this.bulunduguSehirKodu = bulunduguSehirKodu;
    }
```

`toString` metodu ise bir sınıfın alanlarını içerecek şekilde bir metin ifade eder. İhtiyaca göre şekillendirilebilir.

`Universite` sınıfının

```
@Override
    public String toString() {
        return "Üniveritenin;"
            + "adı:" + ad
            + ", kurulusYili=" + kurulusYili
            + ", bulunduguSehirKodu=" + bulunduguSehirKodu
                + "dır.";
    }
```

şeklinde de yazılabilir,

```
@Override
    public String toString() {
```

```
        return "Universite [ad=" + ad + ", kurulusYili=" +
kurulusYili + ", bulunduguSehirKodu=" + bulunduguSehirKodu
                    + "]";
    }
```

şeklinde de yazılabilir.

Önemli olan alanları kapsayan bir metin, bir karakter katarı, bir String döndürmesidir.