BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BLM0121 – Nesneye Yönelik Programlama
Bitirme Sınavı

Ad&Soyad	**: CEVAP ANAHTARI**
Öğrenci Numarası	**:**

Akademik yıl : **2023-2024**

Dönem : **Bahar**

Tarih : **4 Haziran 2024 – 09:00**

Sınav süresi : **90 dakika**

Öğr. görevlisi : **Doç. Dr. Ergün GÜMÜŞ**

Soru	1	2	3	4	5	Toplam
Puan	20	20	20	20	20	100
Not						

KURALLAR

- Sınava başlamadan önce Ad&Soyad ve Öğrenci numarası alanlarını doldurunuz.
- Sınav öncesinde ve süresince sınav gözetmenlerinin tüm uyarılarına uymanız gerekmektedir.
- Sınav öncesinde cep telefonlarınızı KAPATINIZ!
- Soruları yanıtlamak için sadece sınav kâğıdınızla beraber verilen kâğıtları kullanmanız gerekmektedir. Yanıtlarınız açık ve okunaklı olmalıdır.
- Sınav boyunca masanızın üzerinde bulunabilecek malzemeler sadece sınav kâğıdınız, kalem ve silgidir.
- Sınav süresince herhangi bir nedenle birbirinizle konuşmak ve malzeme (silgi, kalem, kâğıt vb.) alışverişi yasaktır.
- Bu kuralların herhangi birine uymamak kopya çekmeye yönelik bir hareket olarak değerlendirilir ve ilgili makamlara bildirilir.

Sorular

1) Aşağıdaki ifadelerin durumuna göre Doğru veya Yanlış alanını “X” sembolüyle işaretleyiniz.

İfade	Doğru	Yanlış
Bir arayüzdeki tüm veri alanları (data fields) “public static abstract” türündedir.		X
Arayüzler miras alınabilir (extend edilebilir).	X	
Soyut sınıflar somut (concrete) metotlar içerebilir.	X	
Bir soyut sınıfın atası concrete (somut) sınıf olabilir.	X	
Soyut sınıflardan nesneler yaratılabilir.		X
Number n = new Integer(2); ifadesi geçerli bir ifadedir.	X	
NullPointerException sınıfı IOException sınıfını miras alır.		X
Sadece bir tek catch ifadesiyle birden fazla ve farklı türden istisnanın yakalanması mümkün değildir.		X
Exception sınıfından istisna simgeleyen nesneler yaratıp fırlatmak mümkündür.	X	
Derleme zamanında, bir nesnenin hangi metodunun kastedildiğine nesnenin gerçek sınıfına (actual class) bakılarak karar verilir.		X

2) Aşağıda verilen Test sınıfını ve kodun çıktısını inceleyiniz.

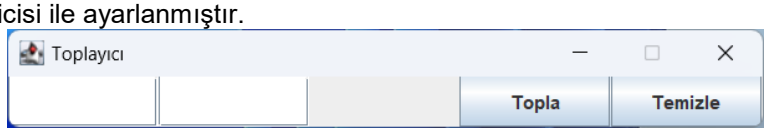
```
import java.util.Arrays;
public class Test {
	public static void main(String[ ] args){
		String[ ] kelimeler =  {"Iki", "Sekiz", "Bir", "Dokuz", "Uc", "Yirmi", "Bir", "Otuz"};
		Metin[ ] metinler = new Metin[kelimeler.length];

		for(int i = 0; i < kelimeler.length; i++)
			metinler[i] = new Metin(kelimeler[i]);

		Arrays.sort(metinler);

		for(int i = 0; i < kelimeler.length; i++)
			System.out.print(metinler[i].m + " ");
	}
}
```

Kodun çıktısı: Uc Bir Bir Iki Otuz Dokuz Sekiz Yirmi

Programın görevi, Metin türündeki “metinler” dizisindeki farklı uzunluktaki kelimeleri, uzunluklarına göre kısadan uzuna doğru sıralamaktır. Aynı uzunluktaki metinler de alfabetik olarak kendi içlerinde sıralanmaktadır. Buna göre Metin sınıfını yazınız.

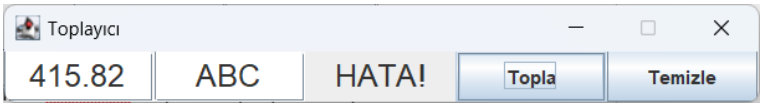
```
class Metin implements Comparable{
	String m;

	Metin(String m) {
		this.m = m;
	}

	@Override
	public int compareTo(Object o) {
		String d = ((Metin)o).m;
		if(m.length() > d.length())
			return 1;
		else if(m.length() < d.length())
			return -1;
		else
			if(m.compareTo(d) > 0)
				return 1;
			else
				return -1;
	}

}
```

3) Aşağıda, sırasıyla, iki adet metin alanı, bir adet etiket alanı ve iki adet düğme içeren bir Swing penceresi görülmektedir. Pencerenin boyutu 500x70 olup içindeki bileşenlerin konumu GridLayout yerleşim düzenleyicisi ile ayarlanmıştır.



"Topla" düğmesinin görevi metin alanlarına girilen rasyonel sayıları toplayarak toplamı etiket alanına yazdırmaktır. "Temizle" düğmesinin görevi de metin alanlarının ve etiketin içeriğini temizlemektir. Aşağıda, programın "Topla" butonuna basıldıktan sonraki ekran görüntüsü görülmektedir.

Eğer toplama işleminde herhangi bir hata (istisna) oluşursa metin alanına aşağıda görüldüğü gibi "HATA!" yazdırılmalıdır.



Buna göre tarif edilen Java programını yazınız.

```
package deneme2;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Pencere extends JFrame implements ActionListener{

	JTextField jtfa, jtfb;
	JLabel jlbl;
	JButton jb1, jb2;

	Pencere(String baslik) {
		super(baslik);
		int en = 500;
		int boy = 70;
		this.setSize(en, boy);
		this.setResizable(false);
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);

		this.setLayout(new GridLayout(1, 5));
		jtfa = new JTextField();
		jtfb = new JTextField();
		jlbl = new JLabel();
		jb1 = new JButton("Topla");
		jb1.addActionListener(this);
		jb2 = new JButton("Temizle");
		jb2.addActionListener(this);

		this.add(jtfa);
		this.add(jtfb);
		this.add(jlbl);
		this.add(jb1);
		this.add(jb2);

		//Font ve hizalama ayarlamaları şart değildir.
		Font f = new Font("Arial", Font.PLAIN, 20);

		jtfa.setFont(f);
		jtfb.setFont(f);
		jlbl.setFont(f);
		jtfa.setHorizontalAlignment(0);
		jtfb.setHorizontalAlignment(0);
```

```
		jlbl.setHorizontalAlignment(0);

		this.setVisible(true);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		JButton kaynak=(JButton)e.getSource();

		if(kaynak == jb2) {
			jtfa.setText("");
			jtfb.setText("");
			jlbl.setText("");
		}
		else {
			try {
				double sayi1 = Double.parseDouble(jtfa.getText());
				double sayi2 = Double.parseDouble(jtfb.getText());
				jlbl.setText("" + (sayi1 + sayi2));
			}
			catch(Exception ex) {
				jlbl.setText("HATA!");
			}
		}
	}
}

public class deneme2 {
	public static void main(String[] args){
		new Pencere("Toplayıcı");
	}
}
```

4) Bu soruda sizden eşzamanlı çalışan 5 görev nesnesi oluşturmanız ve her bir görevi bir thread aracılığıyla çalıştırmanız istenmektedir. Her görev sonsuza kadar çalışacak şekilde tasarlanmalı ve konsola kendi ID numarasıyla beraber A-Z aralığından rasgele seçilen bir harf yazdırarak 1 saniyelik bekleyişe geçmelidir. Programınız yandakine benzer bir çıktı üretmelidir. Buna göre görev sınıfınızı ve test sınıfınızı yazınız.

Program çıktısı

```
Gorev5: H
Gorev1: Z
Gorev3: N
Gorev2: X
Gorev4: L
Gorev1: U
Gorev3: I
…
```

```
package deneme2;

class Gorev implements Runnable{
	int id;

	Gorev(int id){
		this.id = id;
	}

	public void run() {
		char h;
		for(;;) {
			h = (char)('A' + (int)(Math.random() * 26));
			System.out.println("Gorev" + id + ": " + h);
			try {
				Thread.sleep(1000);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}
}

public class deneme2 {
	public static void main(String[] args){
		for(int i = 1; i <= 5; i++)
			new Thread(new Gorev(i)).start();

	}
}
```

5) Aşağıdaki Java terimlerini kısa kod örnekleriyle tanımlayınız.

a) Upcasting / Downcasting [5p]:

```
Integer x = new Integer("1");
Object y = (Object) x; //Upcasting
Integer z = (Integer) y; //Downcasting
```

b) Static/Dynamic Binding [5p]:

```
class A{
	void m() {
		System.out.println("Atadaki metot");
	}
}
class B extends A{
	void m() {
		System.out.println("Yavrudaki metot");
	}
}
public class deneme2{
	public static void main(String[] args) {
		A nesne = new B(); //nesne polimorfiktir.
		nesne.m();
	}
}
```

Derleme zamanında "nesne.m();" ifadesi, "nesne" nin deklare edilen sınıfındaki (A sınıfı) m() metodunu işaret eder. Buna static binding denir. Ancak, çalışma zamanında "nesne" nin gerçek sınıfındaki (B sınıfı) m() metodu çalıştırılır. Buna da dynamic binding denir.

c) "finally" ifadesi [5p]:

```
	static void deneme() throws Exception {
		try {
			int a = 5 / 0;
		}
		catch(Exception e) { throw new Exception("hata olustu!!!"); }

		finally { System.out.println("Exception olsa bile burası çalışır"); }

		System.out.println("Exception varsa burası çalışmaz");
	}
```

finally kullanılmasaydı, Exception oluştuğunda, bu catch ile yakalanır ve catch'den sonraki hiçbir satır çalıştırılmazdı. Ancak finally eklendiğinde, finally bloğu içerisine alınan kod, exception oluşsa bile çalıştırılır. finally'den sonraki hiçbir satır çalıştırılmaz.

d) try with resources [5p]:

```
try(PrintWriter output = new PrintWriter(file); ) {
	output.print("....");
}
catch(Exception e) {
	System.out.println("Bir hata oluştu!");
}
```

Normalde yazım işlemi bittikten sonra output.close(); ile yazma nesnemizi kapatmamız gerekir. Ancak bunu yazmayı unutursak diye try(){…} ifadesinde output nesnemizi () parantezlerinin arasında yaratıyoruz. Böylece Java, biz unutsak bile bunun kapatılması gereken bir kaynak olduğunu bilip işimiz bittiğinde nesneyi otomatik kapatır.