המסלול האקדמי
המכללה למינהל

להצלחה יש דרך

הנדסת תוכנה מוכוונת עצמים

תרגיל 4 – behavioral and structural design patterns

(+ קצת java generics)

ד"ר אליהו חלסצ'י
eliahukh@colman.ac.il

שאלה 1 – עדכונים (40 נק')

ברצוננו ליצור "תכונה" באמצעות המחלקה Property<V>, כאשר V יכול להיות כל טיפוס. נרצה שתהיה לנו האפשרות לכרוך אובייקטים מהסוג של Property כך כאשר האחד משנה את ערכו, כל האובייקטים הכרוכים אליו ישנו אוטומטית את ערכם בהתאמה.

לדוגמא (ראו את ה main הבא):

ניצור את המשתנים msp0, msp1, msp2 מסוג Property<String>. נכרוך את msp1 ל msp0. נכרוך את msp2 ל msp1. כשנשנה את ערכו של msp0 ל "hello world!" אז גם msp1 ישתנה לאותו הערך, ומכיוון ש msp2 כרוך ל msp1 אז גם msp2 ישנה בתורו את ערכו ל "hello world!" (טרנזיטיביות).

```
public static void main(String[] args) {
        Property<String> msp0=new Property<String>();
        Property<String> msp1=new Property<String>();
        Property<String> msp2=new Property<String>();

        msp0.bind(msp1); // when msp0 changes, so does msp1
        msp1.bind(msp2);

        msp0.setValue("hello world!");

        System.out.println(msp2.getValue()); // hello world!
}
```

ב package בשם test ממשו את המחלקה Property<V> כך שה main לעיל יעבוד בהתאם לציפיות (שינוי msp0 גורר שינוי אוטומטי וטרנזיטיבי ל msp2).

- בפרט, עליכם לממש את המתודות bind, setValue ו getValue.
- תשימו לב שע"פ הגדרת השאלה ניתן לכרוך לאובייקט Property כמה Properties שנרצה, לדוגמא נכרוך את msp1, msp2, msp3 ל msp0:
  - msp0.bind(msp1); msp0.bind(msp2); msp0.bind(msp3);
  - כש msp0 ישנה את ערכו לכל השאר יוזן ערך זה אוטומטית.
- הכריכה תהיה אך ורק ל Property עם אותו הטיפוס V.
  - למשל, אם היינו מנסים לכרוך משתנה מסוג Property<Integer> למשתנה מסוג Property<String> אז נרצה לקבל על כך שגיאת קומפילציה.

טיפ: חישבו על design pattern מתאים ומי ממלא איזה תפקיד

שאלה 2 – Command Pattern (20 נק')

ברצוננו לכתוב Controller שמתחזק תור עדיפויות. מצד אחד, במתודה executeOne() אנו רוצים לשלוף פקודה מהתור ולהריץ אותה. לפיכך חשפנו את הממשק:

```
public interface Command {
	public void execute();
}
```

אין לשנות את הממשק הזה.

מצד שני, לא נרצה להגדיר בתוך המחלקה Controller את ה Comparator שבאמצעותו תתבצע ההעדפה בתור העדיפויות. לכן, בבנאי של Controller נבקש Comparator כפרמטר ובאמצעותו נאתחל את תור העדיפויות.

```
public class Controller<_______________> {

	private PriorityQueue<___________> queue;

	public Controller(Comparator<_______________> comparator) {
		queue=new PriorityQueue<_____________>(comparator);
	}

	public void insertCommand(___________ c){
		queue.add(c);
	}

	public void executeOne(){
		if(!queue.isEmpty())
			queue.poll().execute();
	}

}
```

הנה הבנאי שמקבל Comparator כפרמטר

הנה הקריאה ל execute()

לאור האילוצים לעיל, תחילה חישבו מדוע **<u>אי אפשר</u>** ש:

א. המחלקה Controller תוגדר כ Controller<T> ואז תור העדיפויות כ PriorityQueue<T>

ב. תור העדיפויות יוגדר כ PriorityQueue<Command>

<u>טיפ:</u> אם אינכם יודעים את התשובות, השלימו את הקוד לפי שיטות אלה וראו באלו בעיות אתם נתקלים \ כתבו main משלכם כדי להשתמש ב Controller וראו אלו הגבלות נוצרות.

כעת השלימו (אך ורק) את החסר בקוד המחלקה Controller כדי לפתור את הבעיה. שימו לב שרק תשובה נכונה תתקמפל בבדיקה, אם תשובתכם לא התקמפלה היא אינה נכונה.

שאלה 3 - design patterns (20 נק')

נתונה המחלקה הבאה המהווה שאלה אמריקאית. המערך patterns מכיל שמות של design patterns שלמדנו במהלך הסמסטר. בכל מתודה מופיעה דוגמת קוד קטנה בהערה. עליכם להחזיר מחרוזת מתוך מערך patterns (לדוג' patterns[0]) בהתאמה לתבנית אליה משתייכת דוגמת הקוד.

```
public class AmericanQuestion {
	String[] patterns={"Singleton", "Strategy", "Class Adapter", "Object
Adapter", "Decorator", "Observer", "Facade", "Command", "Factory"};
	public String q1(){
		// which design pattern do we see here?
		// new Thread(()->System.out.println("hello world!")).start();
		return null;
	}
	public String q2(){
		// which design pattern does InputStreamReader applies?
		// new BufferedReader(new InputStreamReader(System.in));
		return null;
	}
	public String q3(){
		// which design pattern do we see here?
		// Model m=Model.getInstance();
		return null;
	}
	public String q4(){
		// which design pattern do we see here?
		// (There is a layer named model)
		// Model m=new MyModel();
		return null;
	}
}
```

שאלה 4 – (20 נק')

נתון הממשק הבא:

```
public interface Player {
	public void play();
}
```

וכן נתונה המחלקה הבאה שמימשה אותו:

```
public class MyPlayer implements Player{
	@Override
	public void play() {
		System.out.println("MyPlayer - play()");
	}
	protected void rewind(){
		System.out.println("MyPlayer - rewind()");
	}
	protected void stop(){
		System.out.println("MyPlayer - stop()");
	}
}
```

** **לא ניתן** לשנות את הממשק או המחלקה לעיל.

ב package בשם test, כתבו מחלקה בשם MyAdapter המאפשרת אדפטציה מ MyPlayer ל Runnable כך שנוכל להפעיל את MyPlayer בת'רד נפרד. הפעלה זו תכלול קריאה ל stop() לאחר מכן ל rewind() ולבסוף ל play().

שימו לב שההרשאה של rewind ו stop היא protected, ואף אחד לא מחייב אותנו שהשימוש ב MyAdapter ייעשה באותה ה package של MyPlayer.

תזכורת להרשאות ב Java:

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
public	+	+	+	+	+
protected	+	+	+	+	o
no modifier	+	+	+	o	o
private	+	o	o	o	o

+ : accessible
o : not accessible

הגשה

למערכת הבדיקה הגישו לתרגיל ex3 את הקבצים הבאים, ואותם בלבד:

- Property.java
- Controller.java
- AmericanQuestion.java
- MyAdapter.java

שימו לב שכל המחלקות מוגדרות כבתוך package בשם test

MainTrain

בעיקר נועד לעזור לכם למנוע שגיאות קומפילציה, ולאו דווקא לבדוק מקרי קצה (שדווקא כן ייבדקו ב mainTest).

```
public class MainTrain {

	static class MyCommand implements Command{
		@Override
		public void execute() {/*...*/}
		int getSomePriority(){ /*...*/ return 0; }
	}


	public static void main(String[] args) {
		// 1
		Property<String> msp0=new Property<String>();
		Property<String> msp1=new Property<String>();
		Property<String> msp2=new Property<String>();

		msp0.bind(msp1); // when msp0 changes, so does msp1
		msp1.bind(msp2);

		msp0.setValue("hello world!");

		System.out.println(msp2.getValue()); // hello world!

		// 2
		Controller<MyCommand> c=new Controller<MyCommand>(new
Comparator<MyCommand>() {
			@Override
			public int compare(MyCommand o1, MyCommand o2) {
				return o1.getSomePriority()-o2.getSomePriority();
			}
		});

		c.insertCommand(new MyCommand());
		c.executeOne();

		//3
		System.out.println(new AmericanQuestion().q1());

		//4
		new Thread(new MyAdapter()).start();
	}

}
```