המסלול האקדמי
המכללה למינהל

להצלחה יש דרך

מועד ב' תשע"ז

פיתוח תוכנה מתקדם 1

משך הבחינה 3 שעות. יש לענות על כל השאלות.

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

שאלה 1 – design patterns (35 נק')

סעיף א' (15 נק')

נתונה המחלקה Task הבאה, המגדירה תדירות, משך ושם.

```
package test;

public class Task {

	int frequency;
	int duration;
	String name;

	public Task(int frequency, int duration, String name) {
		this.frequency=frequency;
		this.duration=duration;
		this.name=name;
	}
}
```

עדכנו את המחלקה כך שתהיה "ברת השוואה". כלומר, יהיה ניתן למיין רשימה של Task-ים באופן הבא:

```
LinkedList<Task> tasks=new LinkedList<>();
//...
Collections.sort(tasks);
```

** שימו לב, אין צורך להוסיף Comparator למתודה sort.

בלוגיקת ההשוואה תינתן עדיפות לתדירות **גבוהה** יותר. במידה והתדירות שווה תינתן עדיפות למשך **הקצר** יותר. במידה והמשכים שווים אז תינתן עדיפות לשם **הקצר** יותר לקסיקוגרפית.

סעיף ב' (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.

שאלה 2 – עדכונים (35 נק')

בדומה ל StringProperty של javaFX שראינו בשיעורים, 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 3 – (30 נק') עקרונות תכנות, חשיבה, תכנות גנרי

סעיף א' (20 נק') בכל מתודה מופיע בהערות קוד **שמפר** את אחד מהעקרונות של SOLID או GRASP שלמדנו. תחזירו את המחרוזת מתוך הרשימה (לדוגמא principles[0]) **שמתארת הכי טוב** את העיקרון שהופר. כל שאלה שווה 5 נק'.

```
public class AmericanQuestion {
	// SOLID & GRASP Principles
	String[] principles={	"Single responsibility",	//0
				"Open close",		//1
				"Liskov substitution",	//2
				"Interface segregation",	//3
				"Dependency inversion",	//4
				"Creator",		//5
				"Controller",		//6
				"Pure Fabrication",	//7
				"Information Expert",	//8
				"High Cohesion",	//9
				"Low Coupling",		//10
				"Indirection",		//11
				"Polymorphism",		//12
				"Protected Variations"	//13
			};

/***
 * each question depicts a code in the comments which defies a principle
 * return the string which **best** depicts this principle
 */

	public String q1(){
		// void execute(Runnable r){
		//		...
		//		((MyRunnable)r).setParams(x,y,z);
		//		r.run();
		// }
		return _______________;
	}
	public String q2(){
		// MathematicalObject m=new MathematicalObject();
		// double x = m.cos(2*m.Pi()) + m.sin(m.e());
		return _______________;
	}
	public String q3(){
		// Maze m=new Maze();
		// m.generate(10,10);
		// m.setCharacter(0,0);
		// m.print();
		return _______________;
	}
	public String q4(){
		// User u=new User();
		// Date d=new Date();
		// Event e=new Event();
		// Calendar c=new calendar();
		// u.insertNewEvent(e,c,d);
		return _______________;
	}
}
```

סעיף ב' (10 נק') (הרבה) חשיבה, (וממש מעט) תכנות גנרי.

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 (10- נק').

הגשה

יש להגיש למערכת את **כל הקבצים הבאים ואותם בלבד**. במבחן זה אין צורך במחלקות עזר, אך אם ברצונכם להשתמש במחלקות עזר כתבו אותם בתוך אחת מהמחלקות האלה. עליכם לוודא ששמות הקבצים והמחלקות שבתוכם זהים לרשימה זו, אחרת הקוד לא יתקמפל ויופחתו על כך נקודות. כל המחלקות צריכות להיות מוגדרות ב package בשם test.

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

תוכלו להגיש כמה פעמים שתרצו בזמן המבחן ולקבל משוב האם הקוד שלכם מתקמפל ורץ כהלכה. במידה ולא, עליכם לוודא שהקפדתם על ההוראות ולהגיש מחדש במסגרת זמן המבחן.

5 דק' מתום הבחינה תיסגר האפשרות להגיש את המבחן.

זכרו: בכל הגשה עליכם תמיד להגיש את כל הקבצים לעיל ורק אותם.

פלט אפשרי עשוי להיראות כך:

```
copying test main...TestTask.java
compiling...
your code contains compile errors. points are reduced. please check your code and submit again.

copying test main...TestMyAdapter.java
copying test main...MyPlayer.java
copying test main...Player.java
compiling...
running...
your code is compiled successfully and runs without runtime exceptions.

copying test main...TestProperty.java
compiling...
running...
your code contained runtime exceptions and thus points are reduced. please check your code and submit again.

copying test main...TestAmericanQuestion.java
compiling...
running...
your code is compiled successfully and runs without runtime exceptions.

copying test main...TestController.java
copying test main...Command.java
compiling...
your code contains compile errors. points are reduced. please check your code and submit again.
```

בדוגמא לעיל ל Task יש שגיאת קומפילציה, MyAdapter רץ חלק, Property נתקל בשגיאות בזמן ריצה, AmericanQuestion רץ חלק, ול Controller היתה שגיאת קומפילציה. זכרו שריצה חלקה רק אומרת שהבדיקה לא קרסה, הבדיקה עצמה יכולה כמובן למצוא טעויות נוספות.

לינק להגשה:

http://ck.cs.colman.ac.il/java_test_moed_b_2017.jsp

לינק למערכת הגיבוי:

http://db.cs.colman.ac.il/test

בהצלחה!