המסלול האקדמי
המכללה למינהל

להצלחה יש דרך

הנדסת תוכנה מוכוונת עצמים

תרגיל 3 – creational design patterns

(+ קצת java generics)

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

שאלה 1 – Factory (40 נק')

ברצוננו ליצור את המחלקה GenericFactory<Product> הממשת Factory Pattern כללי ע"פ ההגדרות הבאות:

1. המחלקה תעשה שימוש (א) בממשק הפרטי Creator. שימו לב שבהגדרתו הוא אינו זורק חריגות. (ב) במפה.
2. המתודה insertProduct תקבל (א) מחרוזת המהווה מפתח להשגת המוצר, ו (ב) משתנה c מסוג Class המתאים לכל סוג של Product. בהתאם לתבנית העיצוב, מתודה זו תשייך לכל מוצר c יוצר משלו, ותמפה את היוצר הזה למפתח.
3. בהינתן מחרוזת מפתח, המתודה getNewProduct תחזיר לי אובייקט חדש מהסוג של Product. חזרה זו צריכה להיות ב O(1). במידה והמפתח לא קיים המתודה תחזיר null.

השלימו את הקוד הבא כדי לעמוד בדרישות אלו (המקומות החסרים + 2 המתודות האחרונות).

```
package test;


public class GenericFactory<Product> {

	private interface Creator<_______>{
		public _______ create(); // no unhandled exceptions
	}

	Map<String,_____________> map;

	public GenericFactory(){
		map=new __________<>();
	}

	public void insertProduct(String key, Class<__________> c) {
	}

	public Product getNewProduct(String key){
		return null;
	}
}
```

דוגמא לשימוש במנגנון זה (המנהל, המפתח והארכיטקט הם סוגים של עובד):

```
GenericFactory<Employee> factory=new GenericFactory<Employee>();
factory.insertProduct("manager", Manager.class);
factory.insertProduct("developer", Developer.class);
factory.insertProduct("architect", Architect.class);
//...
Employee e=factory.getNewProduct("manager");
```

שאלה 2 – Builder (40 נק')

ברצוננו ליצור את הטיפוס Goblin כאשר כל אובייקט מטיפוס זה הוא immutable. לכן, כל שדותיו מוגדרים כ final. אולם, רק השם והצבע הם שדות חובה. השלימו את הקוד (מקמות חסרים + מתודות ריקות) ע"פ ה Builder pattern עבור אובייקטים שהם immutable.

```
package test;

public class Goblin {

        private final String name;  // required
        private final String color; // required
        private final int size;          // not required
        private final byte iq;           // not required

        ____________ Goblin(___________ builder) {
        }

        ___________  ___________ class GoblinBuilder{
                private final String name;
                private final String color;
                _________________
                _________________

                _______ GoblinBuilder(_______________________) {
                }

                _______ _____________ withSize(int size){}

                _______ _____________ withIq(byte iq){}

                _______ ________ build(){
                        return _____________;
                }
        }

        public String getName() { return name; }

        public String getColor() { return color;}

        public int getSize() { return size;}

        public byte getIq() { return iq;}

}
```

שימוש לדוגמא:

```
//new Goblin(...); // cannot create goblins without the builder
new Goblin.GoblinBuilder("zoot","green").withSize(500).withIq((byte)40).build();
Goblin g1=new Goblin.GoblinBuilder("danganesh","green").withSize(600).build();
Goblin g2=new Goblin.GoblinBuilder("gilgal","red").withIq((byte)100).build();
```

שאלה 3 – Abstract Factory (20 נק')

עבור הטיפוסים מהסוג של A ו B, השלימו את הקוד הבא כך ש AbstractFactory1 יחזיר מצורים מסוג "1" ואילו AbstractFactory2 יחזיר מוצרים מסוג "2".

```
public class Q3 {

	public abstract class A{ public abstract String get();  }
	public class A1 extends A{ public String get(){return "a1";} }
	public class A2 extends A{ public String get(){return "a2";}  }

	public abstract class B{ public abstract String get(); }
	public class B1 extends B{ public String get(){return "b1";} }
	public class B2 extends B{ public String get(){return "b2";} }

	public interface AbstractFactory{
		A createA();
		B createB();
	}
	// מכאן ואילך עליכם להשלים את הקוד
	public class AbstractFactory1{}

	public class AbstractFactory2{}

}
```

הגשה

עליכם להגיש את:

- GenericFactory.java
- Goblin.java
- Q3.java

למערכת ההגשה תחת השם ex2 בקורס software engineering.

אין להגיש את main האימון (הנמצא בסוף מסמך זה)או כל main אחר.

כל המחלקות צריכות להיות מוגדרות תחת ה package בשם test.

בהצלחה!

MainTrain

```
public class MainTrain {

	public static class A{
		static int count=0;
		public A(){
			count++;
		}
	}
	public static class B extends A{}
	public static class C extends B{}

	public static class Employee{

		@SuppressWarnings("unused")
		private String name;

		public void setName(String name){
			this.name=name;
		}
	}

	public static void main(String[] args) {
		GenericFactory<A> factory=new GenericFactory<A>();
		factory.insertProduct("a", A.class);
		factory.insertProduct("b", B.class);
		factory.insertProduct("c", C.class);

		A a=factory.getNewProduct("a");
		factory.getNewProduct("a");
		if(A.count!=2)
			System.out.println("failure getting the true number of
instances");
		if(!a.getClass().equals(A.class))
			System.out.println("failure creating type A");
	if(!factory.getNewProduct("b").getClass().equals(B.class))
			System.out.println("failure creating type B");
	if(!factory.getNewProduct("c").getClass().equals(C.class))
			System.out.println("failure creating type C");

//Goblin error=new Goblin(...); // cannot create goblins without the builder
// note: the following code checks certain compile errors...
new Goblin.GoblinBuilder("zoot","green").withSize(500).withIq((byte)40).build();

		Q3 q3=new Q3();
		AbstractFactory af=q3.new AbstractFactory1();
		if(!af.createA().get().equals("a1"))
			System.out.println("wrong implementation for AbstractFactory1
createA()");	// -5
		if(!af.createB().get().equals("b1"))
			System.out.println("wrong implementation for AbstractFactory1
createB()");	// -5
		af=q3.new AbstractFactory2();
		if(!af.createA().get().equals("a2"))
			System.out.println("wrong implementation for AbstractFactory2
createA()");	// -5
		if(!af.createB().get().equals("b2"))
			System.out.println("wrong implementation for AbstractFactory2
createB()");	// -5

	}

}
```