OOPf (TDA550) Handledare: Anton Palmqvist
palmqvist.anton(SnabelA)gmail.com

SI-Pass 6
https://github.com/epoxy/SIPass

1)
a)
Skapa en klass som heter *RacingTurtle.* Klassen ska extenda klassen ***Thread***.

b)
Overridea metoden *run().* Vid körning av denna metod ska sköldpaddan gå ett steg framåt. Gör en utskrift vid varje steg som säger att sköldpaddan har gått ett steg samt totala antalet steg som den har gått. (Utskrift av sköldpaddans namn är även lämpligt).

c)
En sköldpadda är i mål efter att ha gått 10 steg. Vid målgång skall utskrift göras i stil med:
<Namn på sköldpadda> finished!

d)
Nu ska ni skapa två skölpaddor som ska få tävla mot varandra:) Skapa en ny Main-klass där ni skapar två sköldpaddor. Därefter *startar* ni dess trådar.

e)
Kör programmet flera gånger och se om ni får samma utskrift. Varför/varför inte får ni det?

f)
Lägg till lämpligt sätt att avbryta körningen av en tråd som tillhör en sköldpadda som gått i mål.

Grattis! Ni har nu skapat ett program som använder parallellism med hjälp av trådar:)

2)
a)
Lägg till kod som gör att sköldpaddan efter varje steg har en 50-procentig risk att somna och sova i 2 sekunder.

b)
Lägg in de nya sov-sköldpaddorna i Main-klassen från uppgift 1. Kör programmet. Hur skiljer sig utskriften och exekveringen från innan?

3)
Betrakta koden nedan:

```
public class HipsterCat {
	private String name;
	private int age;
	private int nbrOfLivesLeft;
	private boolean hasGlasses;
	public HipsterCat(String name, int age, boolean hasGlasses){
		this.name = name;
		this.age = age;
		this.hasGlasses = hasGlasses;
		nbrOfLivesLeft = 9;
	}
	public String getName(){
		return name;
	}
	public int getNbrOfLivesLeft(){
		return nbrOfLivesLeft;
	}
	public void decrementLives(){
		nbrOfLivesLeft--;
	}
}
```

a)
Ni vill nu skapa en ny klass som heter *CrazyHipsterCat* som extendar *HipsterCat*. *CrazyHipsterCat* ska likt sköldpaddorna stödja parallellism. Hur löser man detta?
(Ni får alltså inte ändra något i klassen *HipsterCat*).

b)
Skriv *CrazyHipsterCat*-klassen. Det ska finnas med en run-metod som loopar kod där katten har 20-procents risk att bli av med ett liv. Om katten dör skall utskrift göras och loopen avslutas.

c)
Skriv ett Main-program som skapar och “startar” två eller fler katter. Hur skiljer sig detta Main-program från sköldpaddornas?

4)
Betrakta koden nedan:

```
public class Mom implements Runnable{
	private Kitchen kitchen;
	public Mom(Kitchen kitchen){
		this.kitchen = kitchen;
	}
	@Override
	public void run() {
		for(int i = 0; i<10; i++){
			kitchen.bakeCookies();
			System.out.println("Mom baked cookies");
			try{
				Thread.sleep(500);
			}
			catch(InterruptedException e){
				System.out.println("Mom sleep error");
			}
		}
	}
}
public class Kid implements Runnable{
	private Kitchen kitchen;
	public Kid(Kitchen kitchen){
		this.kitchen = kitchen;
	}
	@Override
	public void run() {
		for(int i = 0; i<10; i++){
			kitchen.eatCookies();
			System.out.println("Kid ate cookies");
			try{
				Thread.sleep(500);
			}
			catch(InterruptedException e){
				System.out.println("Kid sleep error");
			}
		}
	}
}
public class Main {
	public static void main(String[] args){
		Kitchen kitchen = new Kitchen();
		Mom mom = new Mom(kitchen);
		Kid kid = new Kid(kitchen);
		Thread momThread = new Thread(mom);
		Thread kidThread = new Thread(kid);
		momThread.start();
		kidThread.start();
	}
}
```

Handledare: Anton Palmqvist
palmqvist.anton(SnabelA)gmail.com
https://github.com/epoxy/SIPass

a)
Skapa klassen Kitchen med dess två metoder *bakeCookies()* och *eatCookies()*.
Kitchen-klassen måste vara trådsäker.

b)
Skriv om klasserna så att *Mom* och *Kid* istället extendar *Thread*.

c)
Blir utskriften och exekveringen annorlunda med koden från uppgift b?

5)
a)
Diskutera skillnaderna och likheterna på:

- Nästlade klasser
- Statiska medlemsklasser
- Inre klasser
- Lokala klasser
- Anonyma klasser

b)
Hitta kodexempel på begreppen från uppgift a).

6)
a)
Skriv en klass som skriver ut en text till en fil och sparar filen på hårddisken.

b)
Lägg till kod i *TurtleRace*-programmet som sparar varje vinst (namnet på sköldpaddan som går i mål först) i en textfil. Ni kan kalla den skapade textfilen för *highscore.txt*