OOPf (TDA550)

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

SI-Pass 3

https://github.com/epoxy/SIPass

1)

När man designar utefter kontrakt (kontraktbaserad design) är det viktigt att jobba med för- och eftervillkor (pre- och post-conditions).

Börja med att skriva kod för följande metoder i ett miniräknarprogram:

- public int add(int term){}
- public int subtract(int term){}
- public int multiply(int factor){}
- public int divide(int denominator){}

Dela upp er vid bordet så att hälften är programmerare och hälften är kunder som vill köpa programmet.

- Programmerarna vill ha lämpliga pre-conditions för att deras program ska få lämpliga värden så att det kan fungera korrekt.
- Kunderna vill ha lämpliga post-conditions för att försäkra sig om att man får ut önskat resultat.

a)

Skriv inom varje delgrupp vilka villkor ni vill ha.

b)

Diskutera nu på hela bordet och kom fram till pre- och post-conditions som ni alla är överens om.

2)

Skriv i gruppen en valfri klass med lämpliga variabler och metoder.

3)

Vilka egenskaper måste vara uppfyllda för att en klass ska vara icke-muterbar?

4)

a)

Titta tillbaka på klassen ni just skrev i uppgift 2. Kontrollera om den uppfyller de icke-muterbara egenskaperna som ni hittat i uppgift 3.

b)

Om inte, ändra/skriv om klassen så att den blir icke-muterbar.

5)
Titta på klassen nedan:

```
public class RollingBall{
	private int radius;
	private double speedOverGround;
	private Date timeBallStartedToRoll;
	public RollingBall(int radius, double speedOverGround, Date
timeBallStartedToRoll){
		this.radius = radius;
		this.speedOverGround = speedOverGround;
		this.timeBallStartedToRoll = timeBallStartedToRoll;
	}
	public int getRadius(){
		return radius;
	}
	public double getSpeedOverGround(){
		return speedOverGround;
	}
	public double getTimeBallStartedToRoll(){
		return timeBallStartedToRoll;
	}
}
```

a)
Varför är denna klass muterbar?

b)
Antag att vi nu vill skapa en flygande boll. Den flygande boll-klass måste dock vara icke-muterbar. Hur kan man lösa detta?

c)
Skriv koden för den flygande boll-klassen.

d)
Lägg till en korrekt equals-metod i er flygande boll-klass.

e)
Lägg till en korrekt hashCode-metod i er flygande boll-klass.

6)
a)
Skapa ett eget Exception med namnet *FasterThanSpeedOfLightException*.

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

b)
Lägg nu till en kontroll i konstruktorn i RollingBall eller FlyingBall som kollar om hastigheten är över ljusets hastighet och i så fall kastar ert exception från a-uppgiften.

c)
Lägg till ett catchblock som fångar upp ert Exception.

d)
När är det bra att använda catch?

e)
När är det dåligt att använda catch?

7)
a)
Diskutera skillnaderna mellan Comparator och Comparable.

b)
När är den ena att föredra framför den andra?