SI-pass 2

Johan Brook och Jesper Persson

11 september 2012

1. “Sant eller falskt?”

Vi börjar med lite mindfuck! Vad skrivs ut i koden nedan? (notera att vi inte rekommenderar er att skriva kod såhär).

```
if (true && (false || true)){
	System.out.println("Yay 1");
}

/* ----------------------- */

if (!true && !(true && false)) {
	System.out.println("Yay 2");
} else {
	System.out.println("Nay 2");
}

/* ----------------------- */

int var = 9;

if ( !(5 > 10) && !(40 % 10 == 0)) {
	System.out.println("Yay 3");
}
else if (10 == var++){
	System.out.println("Yay 3.5");
}
else {
	System.out.println("Nay 3");
}

/* ----------------------- */

int herp = 1;

if (herp++ == 2) {
	System.out.println("Yay 4");
}
else if (++herp == 3){
	System.out.println("Yay 4.5");
}
else {
	System.out.println("Nay 4");
}




/* ----------------------- */

int derp = 10;

if (derp == 11)
	System.out.println("Yay 5");
```

```
        System.out.println("Yay 6");
```

2. Switching the Case

Förutom if-else-block finns det i Java **switch-case**-block (existerar även i andra språk). Försök göra om följande kod till att använda switch-case istället för if-block.

```
char key = 'f';

if (key == 'a') {
        System.out.println("Walk left");
}
else if (key == 'd') {
        System.out.println("Walk right");
}
else if (key == 's') {
        System.out.println("Walk down");
}
else if (key == 'w') {
        System.out.println("Walk up");
}
else {
        System.out.println("Stand still");
}
```

Diskutera! När bör man använda switch-case istället för if-else? Vilka datatyper kan switch-satsen användas för?

3. Jämförelser

Vad skrivs ut nedan, och varför?

```
String a = "Johan gillar hästar";
String b = "Johan gillar hästar";
String c = a;
String d = new String("Johan gillar hästar");

int e = 1;
double f = 1;
double g = 1.00000000;
Integer h = 1;
Integer i = new Integer(1);

if (a == b) {
        System.out.println("Sant 1");
}

if (a == c) {
        System.out.println("Sant 2");
}



if (a == d) {
        System.out.println("Sant 3");
}

if (e == f) {
        System.out.println("Sant 4");
}

if (f == g) {
        System.out.println("Sant 5");
}
```

```
if (e == h) {
	System.out.println("Sant 6");
}

if (h == i) {
	System.out.println("Sant 7");
}
```

4. Variabler, objekt, och referenser

Kolla in koden nedan:

```
1.	int a;
2.	a = 5;
3.	int b = a;
```

Se sedan skissen nedan på hur kodraderna exekveras, en efter en:

Bra illustrerat va? Betrakta nu följande kod:

```
1.	Car myCar;
2.	myCar = new Car();
3.	Car yourCar = myCar;
```

Gör en liknande illustration över exekveringen av den senare kodsnutten! Vad innehåller de båda `Car`-variablerna?

Diskutera! Vad är skillnaden i hur datorn hanterar de båda kodsnuttarna? Vad är viktigt att tänka på när man arbetar med klasstyper?

5. Att bygga klasser – representera saker

Klasser representerar ofta en *sak* – det kan vara ett verkligt objekt eller något mer abstrakt som endast existerar i ditt program. På föreläsningen har ni sett hur en klass består av *tillstånd* och *beteenden*. Nedanför ser ni exempel på hur en klass för en bil, *Car*, kan vara uppbyggd.

```
class Car {

	/* Instansvariabler */

	// Bilens aktuella hastighet
	int speed;

	// Bilens färg representerad med klassen Color
	Color color;

	// Bilens märke
	String brand;


	/*
		Getters/setters – kommandon och frågor för en Car
		I metoderna nedan har vi skrivit med implementationen för att
		ni ska se hur det vanligtvis ser ut.
	*/

	public int getSpeed() {
		return speed;
	}

	public void setSpeed(int newSpeed) {
		speed = newSpeed;
	}

	public Color getColor() {
		return color;
	}

	public void setColor(Color newColor) {
		color = newColor;
	}

	public String getBrand() {
		return brand;
	}

	/* Beteenden (saknar här implementation) */

	public void turnLeft() {
		// Bilen svänger vänster
	}

	public void turnRight() {
		// Bilen svänger höger
	}

	public void honk() {
		// Bilen tutar
	}

	public boolean isMoving() {
		// Sant om bilen rör på sig, annars falskt
	}
```

```
        // etc ..
    }
```

a)

Att göra: Följ instruktionerna nedan och skriv koden genom att utgå från specifikationen av klassen Car.

1. **Skapa** en ny `Car`.
2. **Om** ditt Car-objekt inte rör på sig, sätt hastigheten till 70. **Annars**, skriv ut hastigheten, bilens färg och märke till kommandoprompten, och säg sedan åt bilen att svänga höger.
3. **Skapa** ännu en `Car`.
4. **Om** båda bilarna rör på sig, gör så att båda tutar.

b)

Betrakta klassen för ett Shotgun nedan. Fyll i klassens tänkbara tillstånd/egenskaper (variabler) och beteenden (metoder). Observera att ni inte behöver fylla i klassens **implementation**, dvs. fyll exempelvis inte i koden för de olika metoderna. Se exemplet `Car` i deluppgift a).

```
    class Shotgun {

        /* Fyll i tillstånd/variabler här */

        /* Fyll i beteenden här */


    }
```

Diskutera: Under arbetet, filosofera över vad ett Shotgun egentligen har för egenskaper, vilken data man kan tänkas vilja ha reda på, och vad det kan göra. Det finns oftast inga rätta svar!

6. Dåligt skriven kod

Att hålla sig till både allmänna kodkonventioner och Javas egna är oerhört viktigt. Det är mycket sällan kod läses och underhålls av endast en person (dig). Medarbetare (och Erland på tentan) måste kunna läsa och förstå kod effektivt.

Därför bör klasser, variabler, och metoder ha beskrivande namn. Nedan ser ni ett simpelt program som skriver ut text beroende på om två nummers kvadrater är lika (om programmet överhuvudtaget behövs är en annan fråga …).

Att göra: Markera och ändra de ställen där vi inte har följt Javas kodkonventioner. På vilka ställen är koden så dålig att den inte ens går att kompilera? Vad kan ni effektivisera?

```
class badcode {

    public static void main(String[] args) {

        int Three;
        Three = 3;

        int 1square = Numbertimesitself(Three);
        int ANumber = Numbertimesitself(10);

        isThe_Numbers_Equal(1square, ANumber);
    }


    public static int Numbertimesitself(int z) {
        int f = z;
        int s = f * z;

        return s;
```

```
	}



	public static void isThe_Numbers_Equal(int förstasiffranattkolla,
					int andrasiffranattkolla) {
		boolean true = false;

		if (förstasiffranattkolla == andrasiffranattkolla) {
			true = true;
		}

		if (true == true) {
			System.out.println("Three equals ANumber!");
		}
		else {
			System.out.println("Three does not equal ANumber!");
		}
	}

}
```