EDAA20
Programmering och databaser

Föreläsning 5 – Objekt

2023-09-11, Niklas Fors

Läsvecka 3

	Måndag	Tisdag	Onsdag	Torsdag	Fredag
Programmering	Föreläsning	Föreläsning	Labb 4		Labb 5
Databaser				Databasseminarium	

Anmäl er till seminarietid

Se inspelad föreläsning 2
Förbered deluppgift 1 (ER-modell)

Några får redovisa deras lösningar

Program med objekt

```
public class DrawSquare {
  public static void main(String[] args) {
    SimpleWindow w = new SimpleWindow(600, 600, "Square");
    Square sq = new Square(20, 10, 40);
    sq.draw(w);
    System.out.println(sq.getArea());
  }
}
```

Programmet skapar och använder två objekt:

- Ett `SimpleWindow`-objekt som beskriver ett fönster på skärmen som man kan rita i
- Ett `Square`-objekt som beskriver en kvadrat

Fler exempel på objekt

Inläsning från användaren:

```
Scanner scan = new Scanner(System.in);
double nbr1 = scan.nextDouble();
```

Slumptalsgenerator:

```
Random rand = new Random();
int dots = rand.nextInt(6) + 1;
```

Objektorienterad Programmering

- Java stödjer ***Objektorienterad programmering*** **(OOP)**
- Programmeringsparadigm från 1960-talet
- OOP används ofta i större system

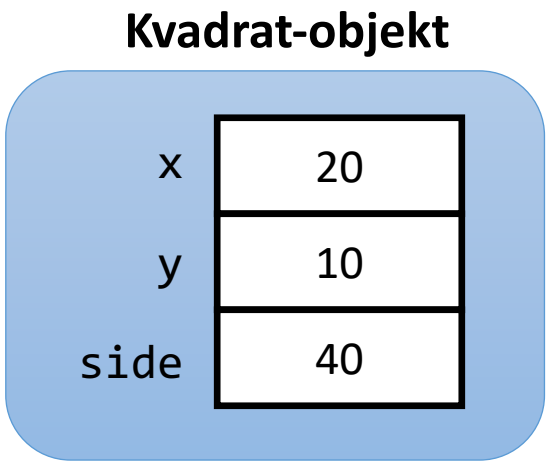
- En ***klass*** definierar **nya *sorters* objekt**
  - Fungerar som en mall/ritning
  - Flera objekt kan skapas av samma klass

Objekt

Ett ***objekt*** har ***attribut*** *(egenskaper/tillstånd)* och ***metoder***

Exempel:

- Kvadratens attribut:
  - Position (x, y)
  - Sidlängd
- Kvadratens metoder:
  - Rita kvadraten i ett fönster
  - Hämta kvadratens area
  - …



Klass

En ***klass*** definierar hur objekt kan skapas och användas

En klass definierar:

- ***Attribut*** - objektets variabler
- ***Metoder***

Skapa objekt

Objekt ***skapas*** med operatorn **new**:

```
new Klassnamn(noll eller flera argument)
```

Exempel:

```
new Square(20, 10, 40)
```

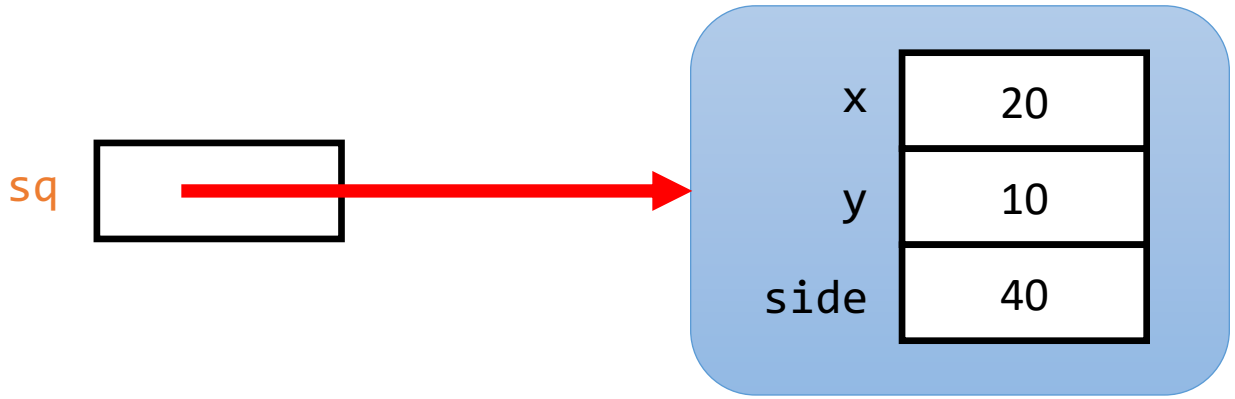
Antal argument och argumentens typer framgår av klassens dokumentation:

Constructor and Description
Square(int x, int y, int side) Skapar en kvadrat med övre, vänstra hörnet i x,y och med sidlängden side.

Referenser

- När objekt skapas får man en **referens** till objektet
- Referenser lagras i ***referensvariabler***

```
Square sq = new Square(20, 10, 40);
```



Referensvariabeln **sq** refererar till ett kvadratobjekt och kan användas för att komma åt objektet (anropa metoder osv)

Metodanrop

Anropa metod på objekt:

```
referens.metodnamn(noll eller flera argument)
```

Exempel:

```
Square sq = new Square(20, 10, 40);
sq.move(30, 10);
int a = sq.getArea();
```

Hur vet man vilka metoder som går att anropa?

Dokumentation för klassen Square:

Modifier and Type	Method	Description
void	**draw**(**SimpleWindow** w)	Ritar kvadraten i fönstret w.
void	**erase**(**SimpleWindow** w)	Raderar bilden av kvadraten i fönstret w.
int	**getArea**()	Tar reda på kvadratens area.
int	**getSide**()	Tar reda på kvadratens sida.
int	**getX**()	Tar reda på x-koordinaten för kvadratens läge.
int	**getY**()	Tar reda på y-koordinaten för kvadratens läge.
void	**move**(int dx, int dy)	Flyttar kvadraten avståndet dx i x-led, dy i y-led.
void	**setSide**(int side)	Ändra kvadratens sida till side.

Metoder har en returtyp, ett namn, och noll eller flera parametrar.

Övning – hitta fel

I följande kod finns tre fel. Hitta felen och rätta dem:

```
Square sq = new Square(20, 10);
sq.draw(w);
SimpleWindow w = new SimpleWindow(600, 600, "Square");
sq.move(30, 10);
sq.draw(w);
System.out.println("Arean: " + w.getArea());
```

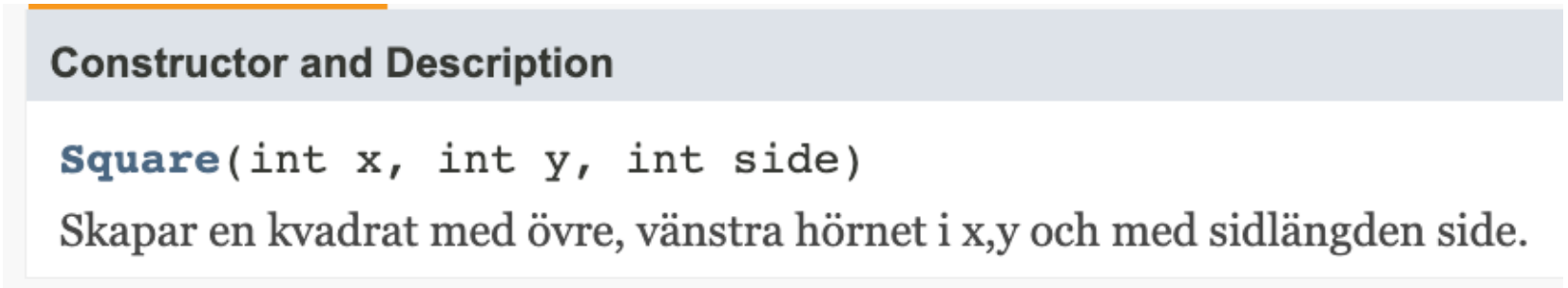
Övning – hitta fel

I följande kod finns tre fel. Hitta felen och rätta dem:

```
Square sq = new Square(20, 10);     // 3 argument behövs
sq.draw(w);                         // w är ej deklarerad än
SimpleWindow w = new SimpleWindow(600, 600, "Square");
sq.move(30, 10);
sq.draw(w);
System.out.println("Arean: " + w.getArea()); // bör vara sq
```

Hitta fel vid anrop

- **Anropas metoden på rätt objekt?** Studera anropet `w.getArea()`
  - Vad har referensen för typ? `SimpleWindow`
  - Finns metoden `getArea` i klassen `SimpleWindow`? Nej!
- **Stämmer argumenten?** Studera **`new`** `Square(20, 10)`
  - Stämmer antal argument? Nej! 3 argument behövs
  - Stämmer typerna? Ja, argumenten kräver typen `int`
  - Stämmer argumentens ordning? Svårt att veta i detta fall



Constructor and Description
`Square``(int x, int y, int side)` Skapar en kvadrat med övre, vänstra hörnet i x,y och med sidlängden side.

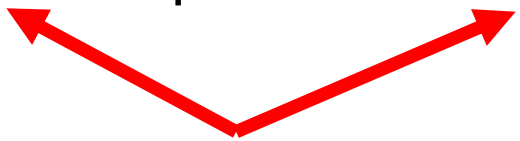
Referensvariabelns typ

Referensvariabelns typ bestämmer vilka sorters objekt variabeln får referera till.

Exempel:

```
Square sq = new Square(20, 10, 40);

Square sq = new SimpleWindow(600, 600, "Square");
```



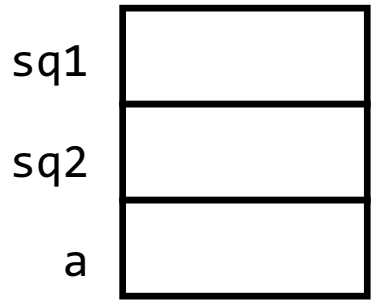
Typerna ej kompatibla
⟹ kompileringsfel!

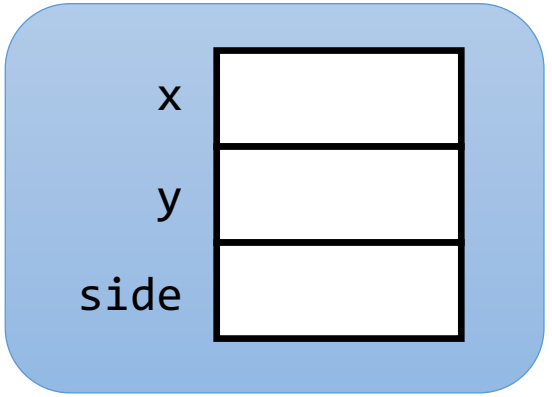
Övning – referenstilldelning

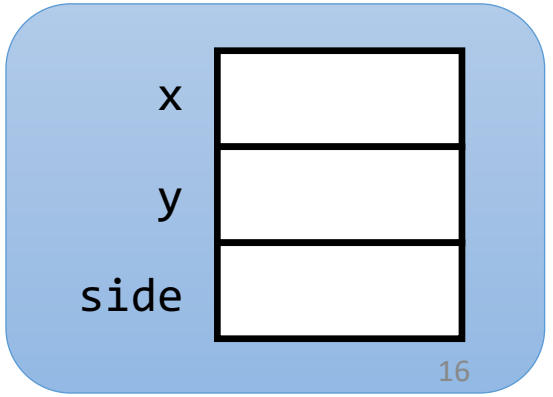
Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```

move flyttar kvadraten relativt





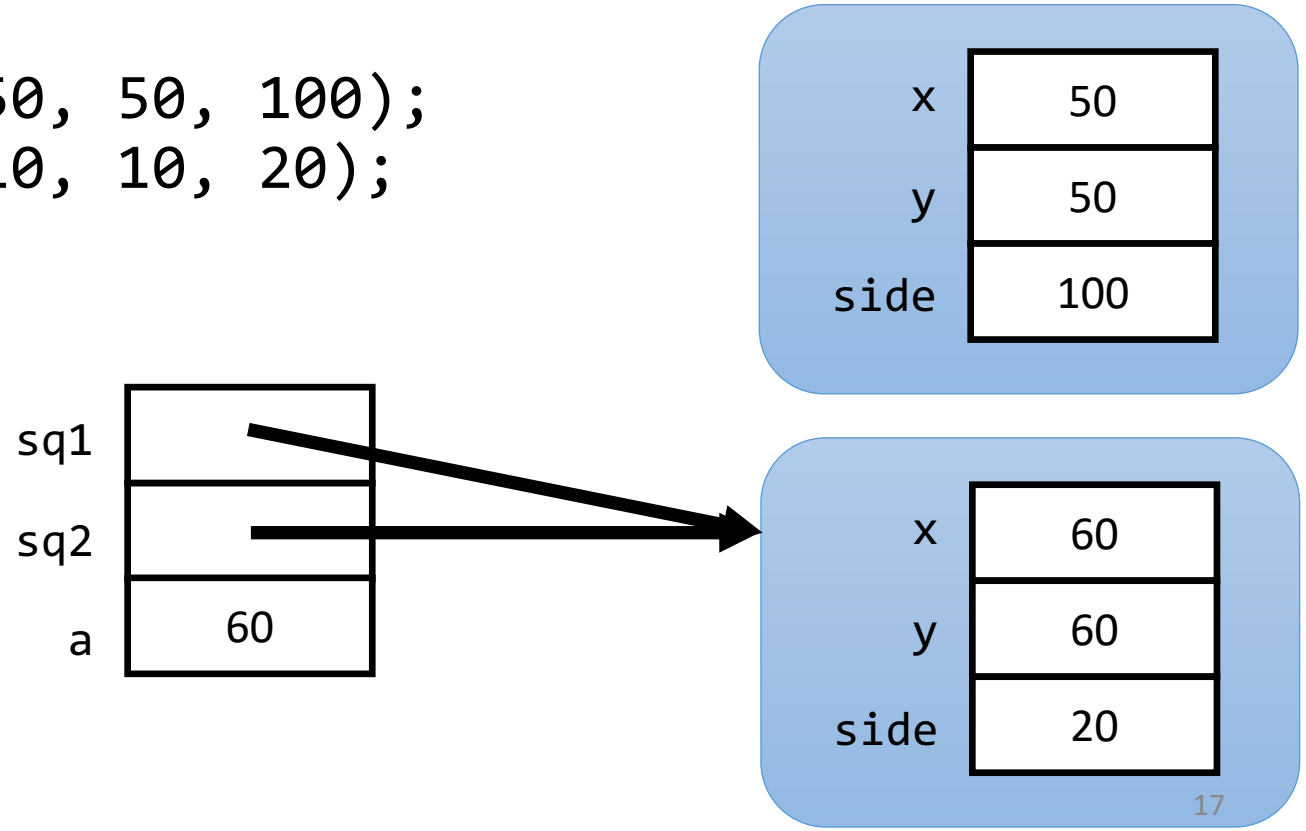


Övning – lösning

Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```

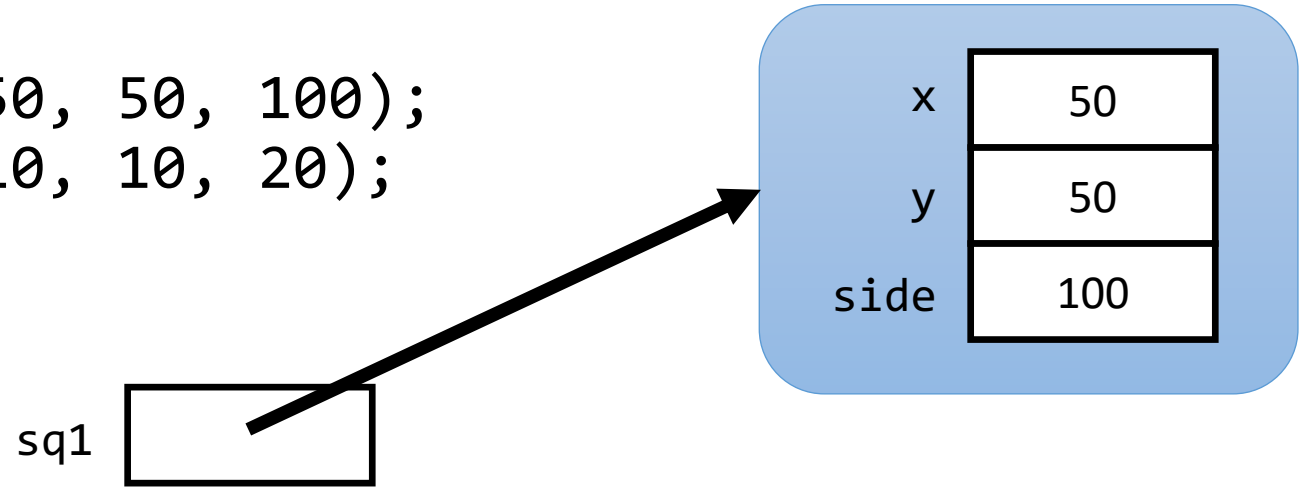
move flyttar kvadraten relativt



Övning – exekvering 1/5

Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

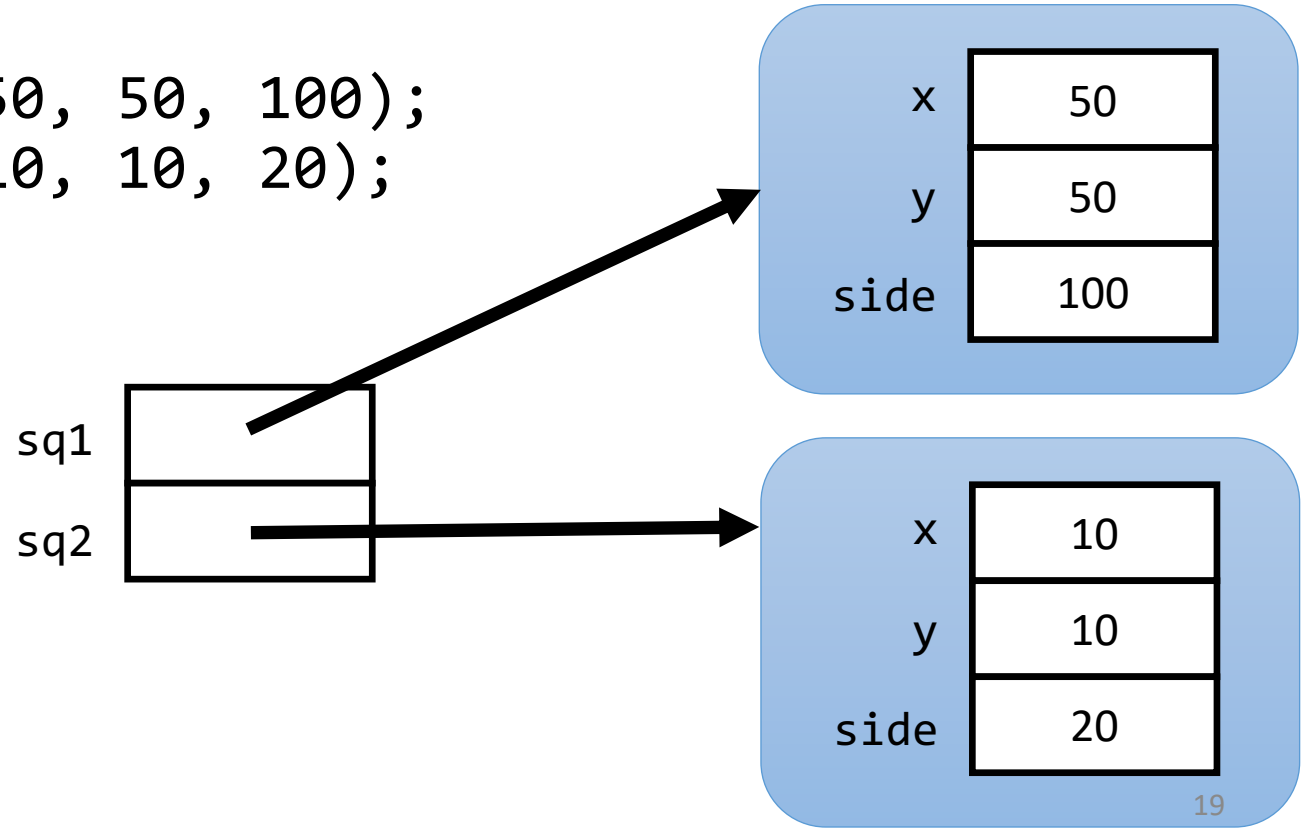
```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```



Övning – exekvering 2/5

Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

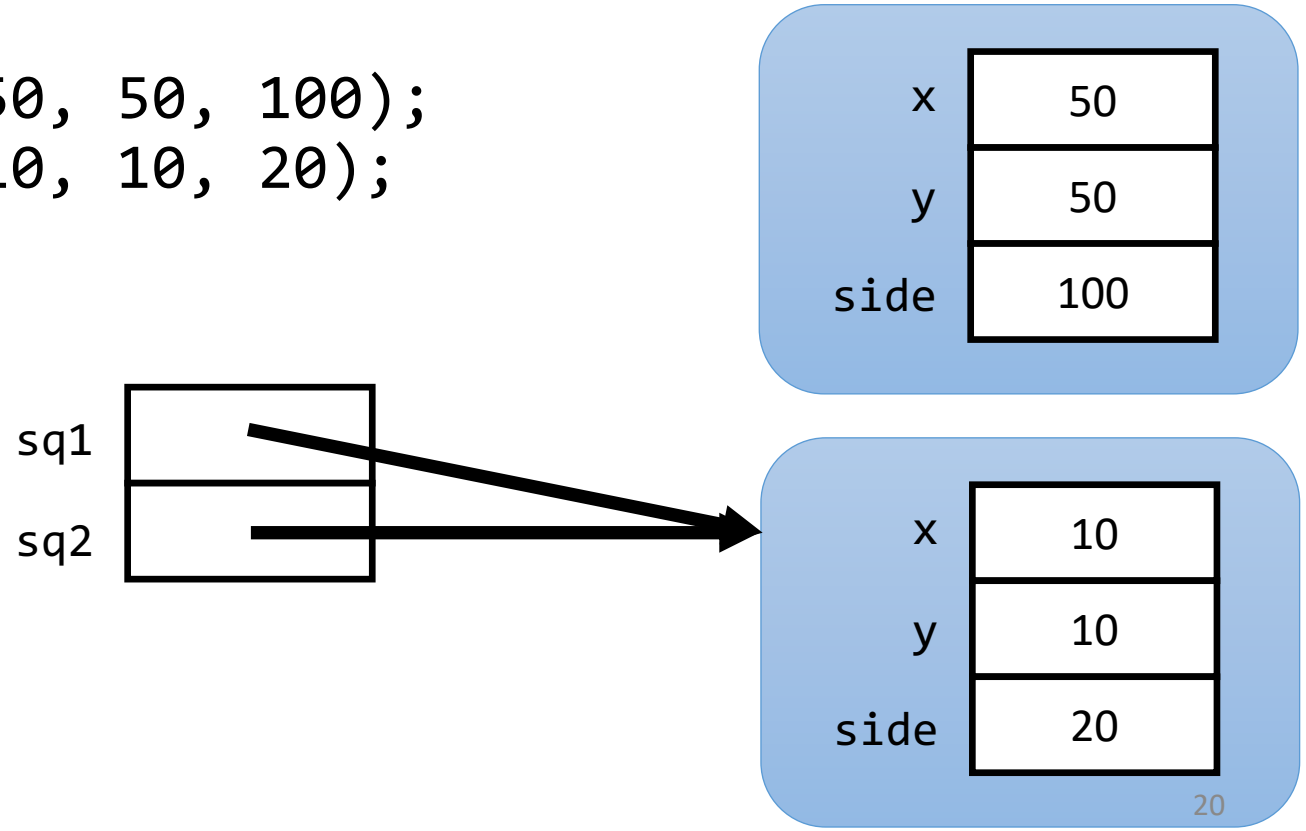
```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```



Övning – exekvering 3/5

Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```

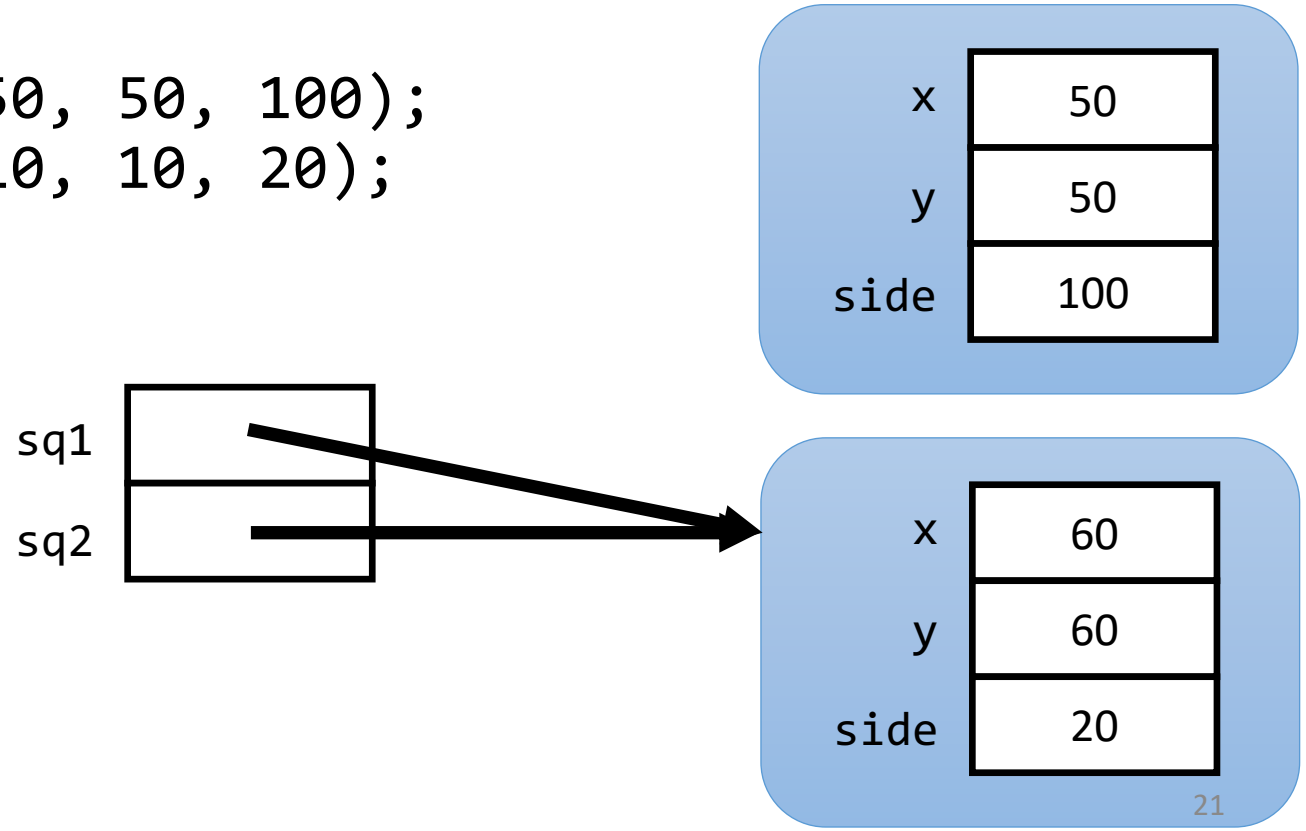


Övning – exekvering 4/5

Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```

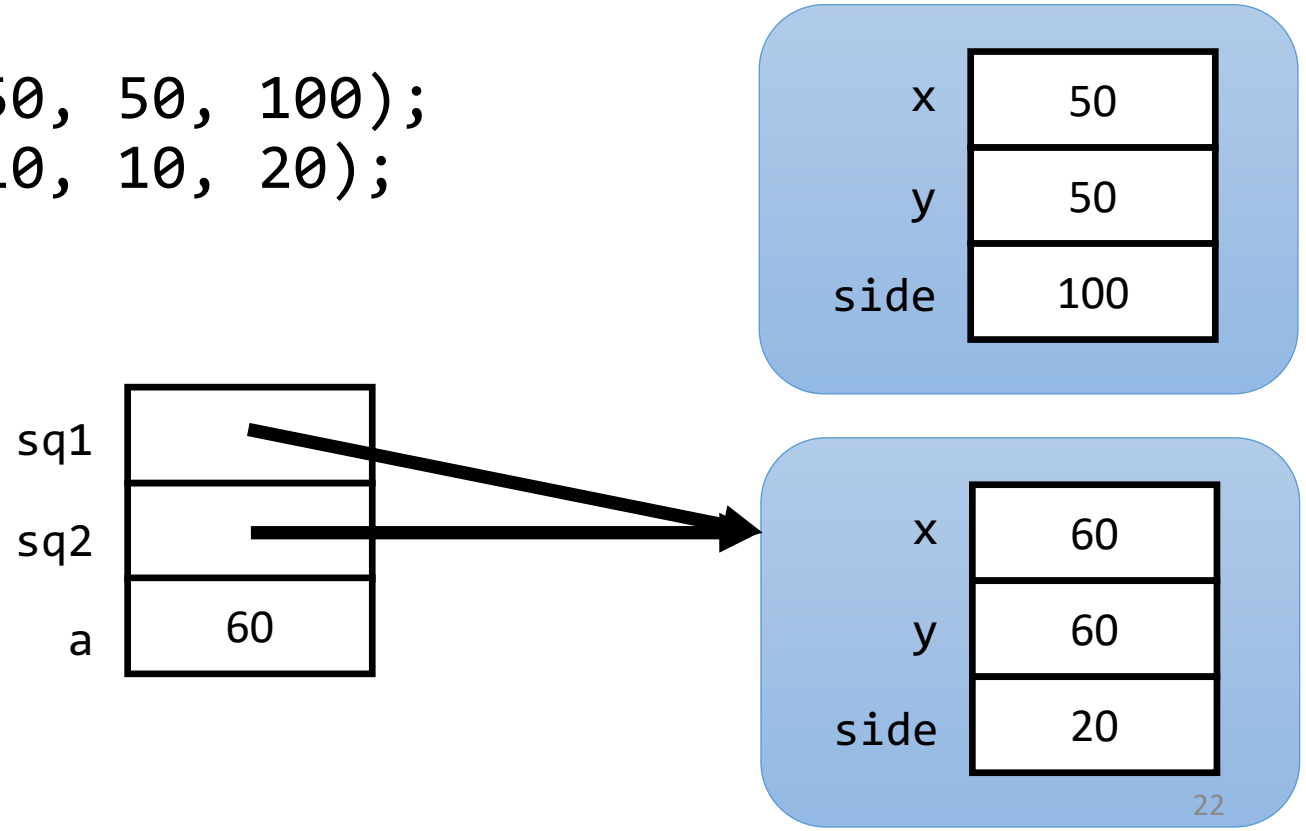
move flyttar kvadraten relativt (10+50=60)



Övning – exekvering 5/5

Fyll i värden (pilar för referenser) på variabler och attribut efter att följande satser har exekverats:

```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```

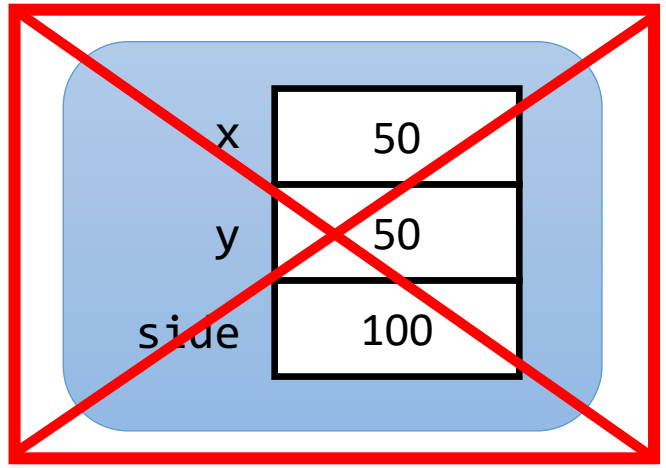


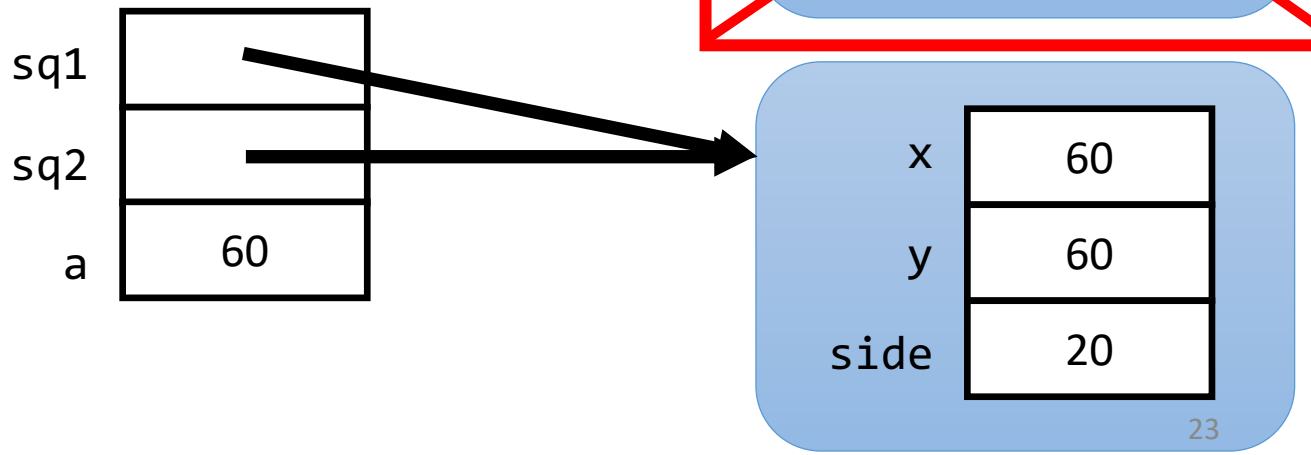
Övning

Fyll i värden …

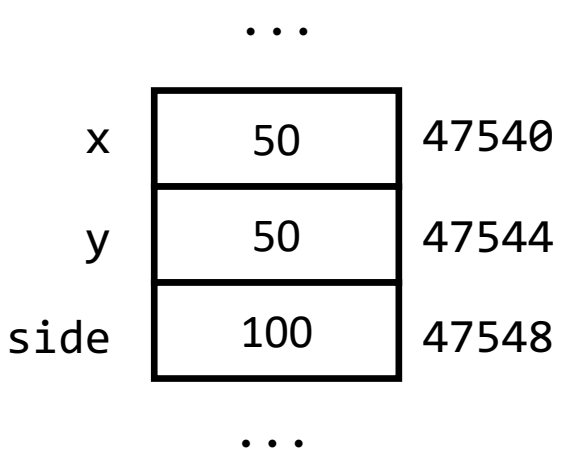
```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
sq1 = sq2;
sq1.move(50, 50);
int a = sq2.getX();
```

Ingen referens finns längre till objektet
⟹ objektet tas bort av Java-systemet
(*garbage collection*)





Minne

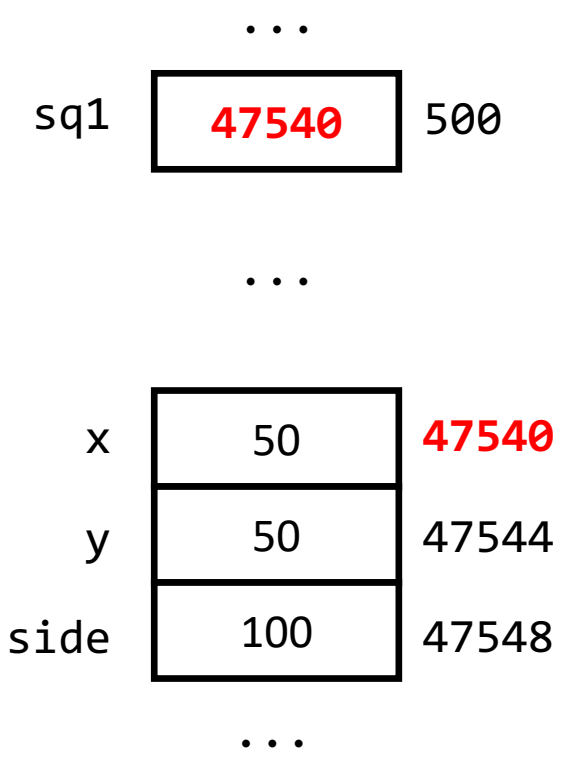


När ett objekt skapas reserveras det plats för attributen i datorns minne.

Exempel:

```
new Square(50, 50, 100)
```

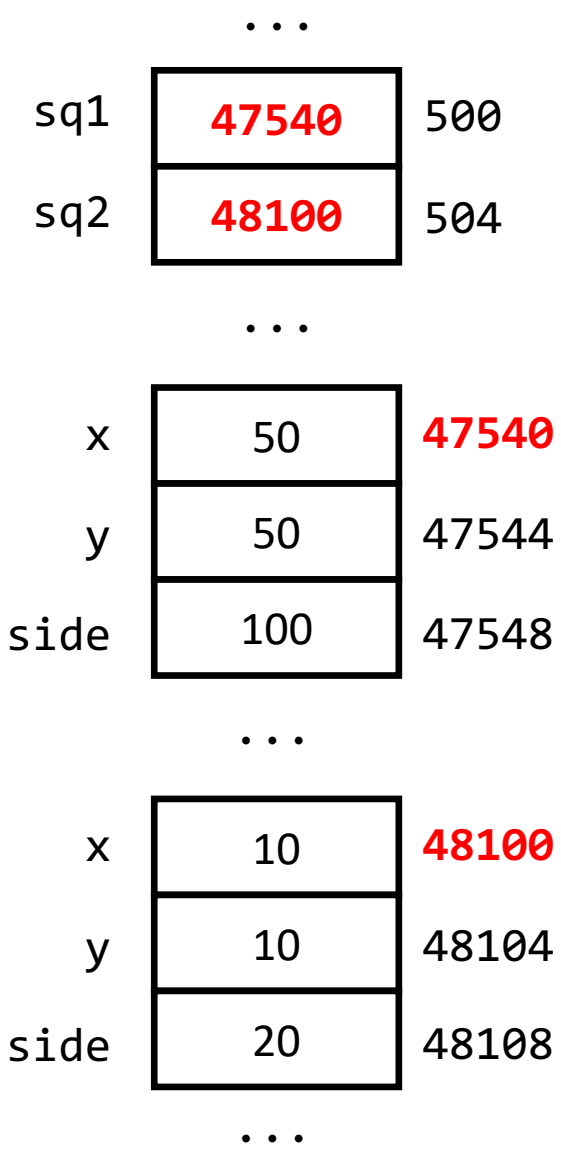
Numren på minnescellerna (adresserna) är bara exempel och inget man behöver tänka på



En referens är adressen till objektet – ett tal

Exempel:

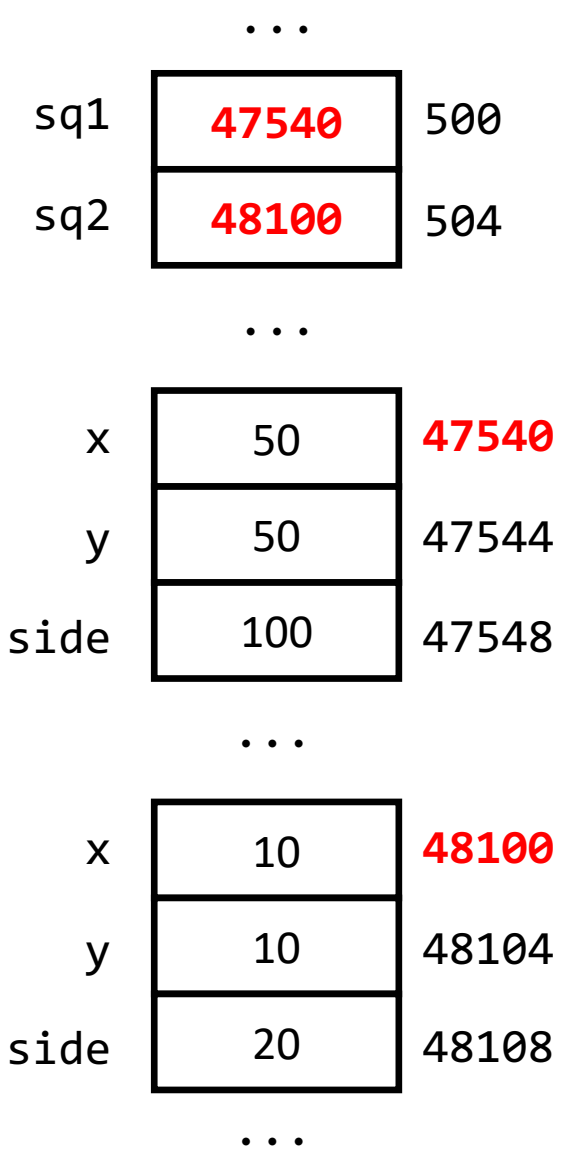
```
Square sq1 = new Square(50, 50, 100);
```



Föregående övning:

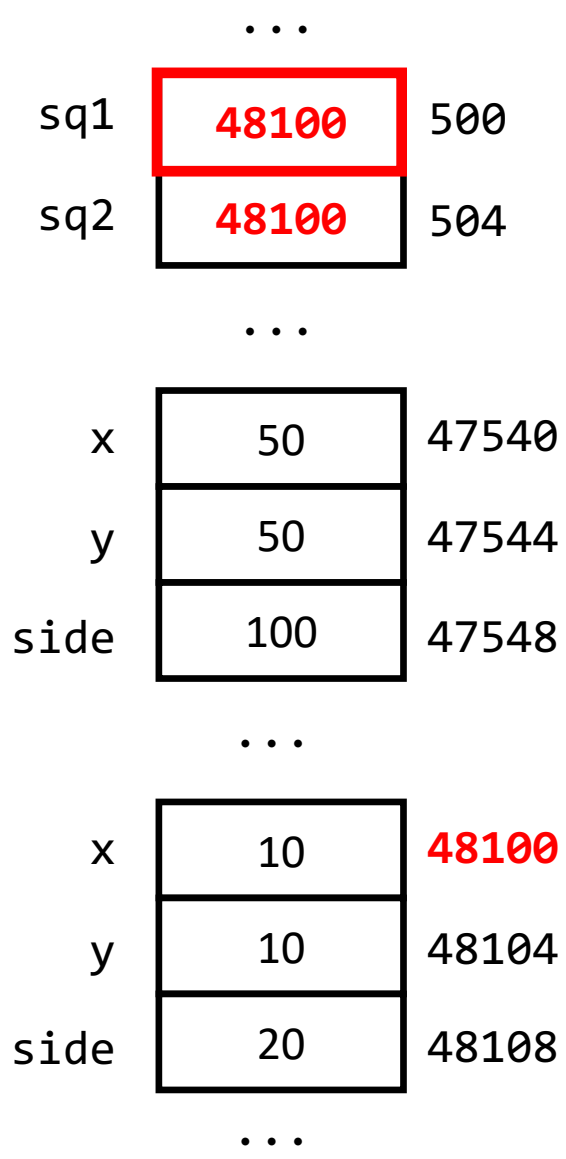
```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(10, 10, 20);
```

Referenstilldelningen i minnet



Efter satsen:

```
sq1 = sq2;
```



Referenstilldelning

När en referensvariabel `sq1` tilldelas värdet av `sq2` är det *adressen* i `sq2` som kopieras till `sq1`. Efter

```
sq1 = sq2;
```

innehåller både `sq1` och `sq2` samma adress, dvs, refererar till samma objekt.

Övning – Vad skrivs ut?

Vad skrivs ut när följande satser exekverats?

```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(50, 50, 100);
System.out.println(sq1 == sq2);
sq1 = sq2;
System.out.println(sq1 == sq2);
```

Vid jämförelse av referensvariabler är det adresserna som jämförs.

Övning – Vad skrivs ut?

Vad skrivs ut när följande satser exekverats?

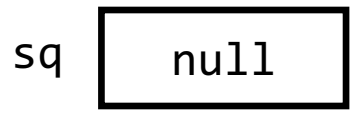
```
Square sq1 = new Square(50, 50, 100);
Square sq2 = new Square(50, 50, 100);
System.out.println(sq1 == sq2);          // false skrivs ut
sq1 = sq2;
System.out.println(sq1 == sq2);          // true skrivs ut
```

Vid jämförelse av referensvariabler är det adresserna som jämförs.

Referensvärdet `null`

En referensvariabel kan tilldelas värdet **`null`**, vilket betyder att variabeln inte refererar till något objekt

```
Square sq = null;
```

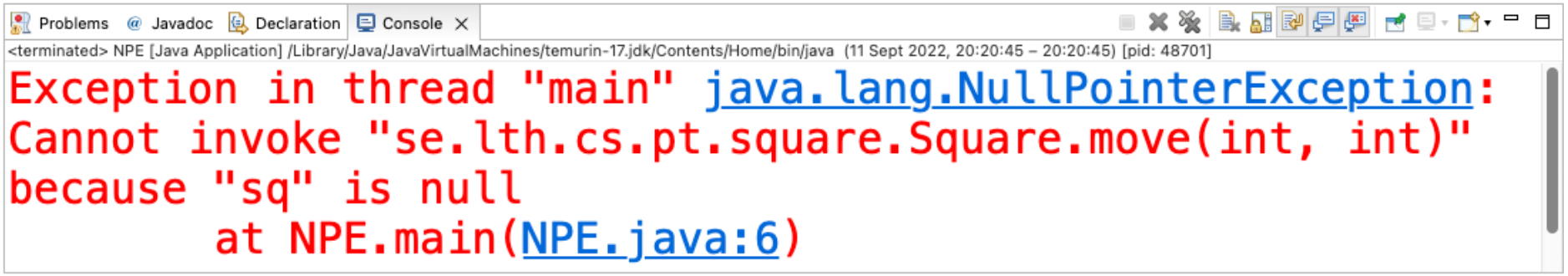


I vissa fall är detta användbart, exempelvis om objektet ej finns än

Vanligt fel: `NullPointerException`

Anropar man en metod på en referensvariabel som är `null` får man exekveringsfel:

```
public class NPE {
	public static void main(String args[]) {
		Square sq = null;
		sq.move(10, 20);
	}
}
```



Viktigt att komma ihåg

- Objekt består av data/egenskaper (attribut) och har metoder
- Objekt skapas genom **`new`** `Klassnamn(...)`
- Referenser till objekt sparas i referensvariabler, vars typ är en klass
- Metoder anropas genom ***`referens.metod(argument)`***
- Värdet av en referensvariabel är en adress till ett objekt
  - Tilldelning (=) betyder att adressen kopieras
  - Jämförelse (==) betyder att adresserna jämförs
- Dokumentationen beskriver hur objekt skapas och vilka metoder som kan anropas