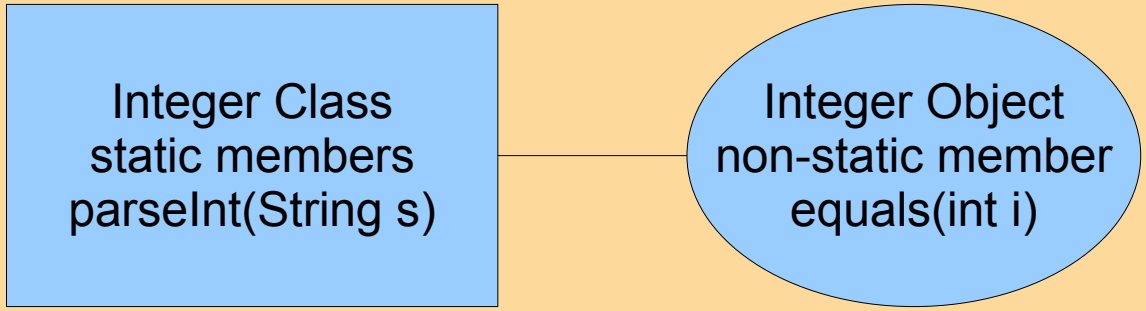
Programming in Java: lecture 6

- Objects, Classes and Instances
- Getters and setters
- Constructors and object initialization
- Wrapper Classes and Autoboxing
- Garbage collection and the heap
- Object oriented analysis and design
- Example

Slides made for use with ”Introuction to Programming Using Java, Version 5.0” by David J. Eck
Some figures are taken from ”Introuction to Programming Using Java, Version 5.0” by David J. Eck
Lecture 3 covers Section 5.1 to 5.4

Classes and Objects

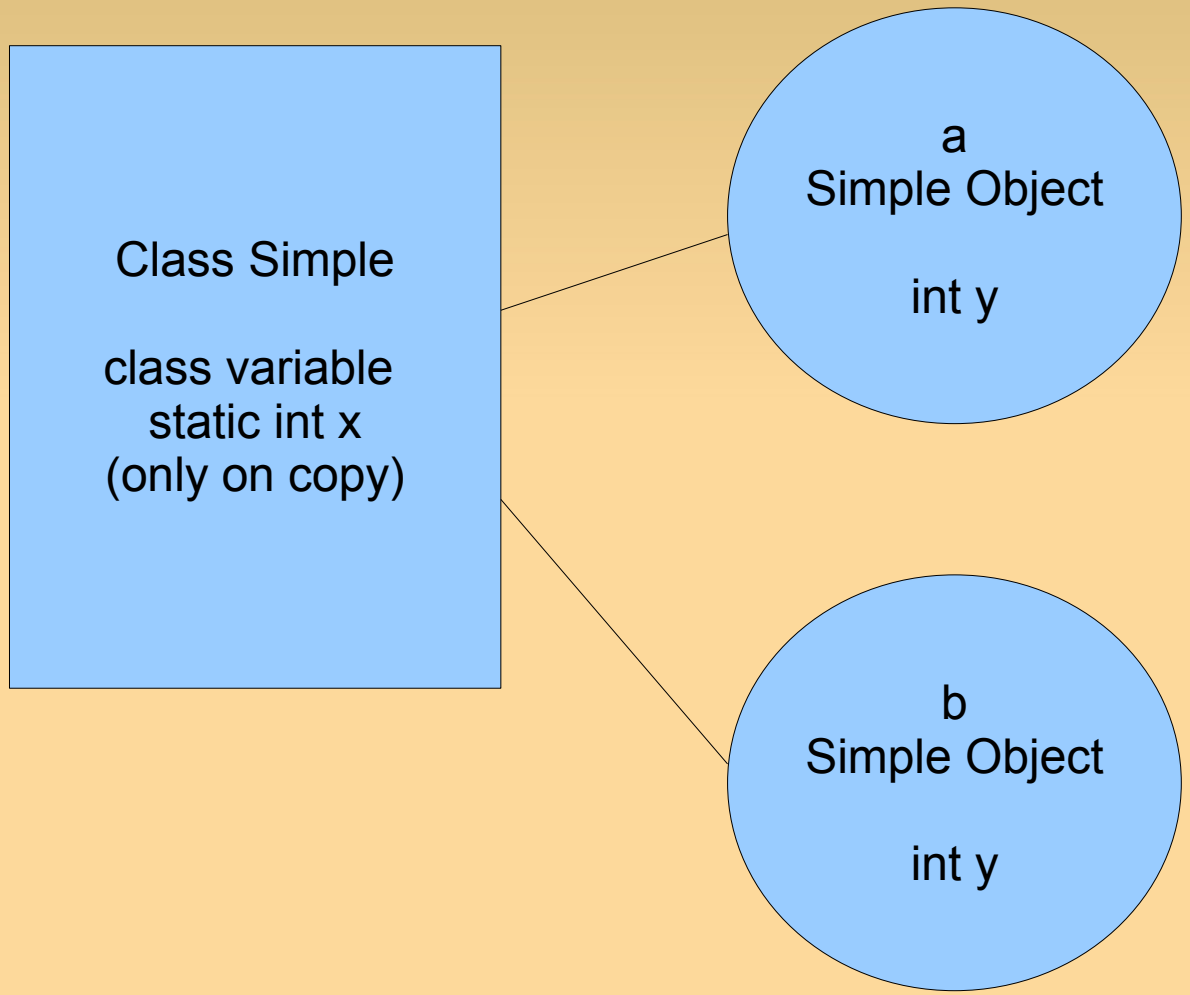
- A Class is a template
- Objects are objects
- Objects are instances of a given class



Instances

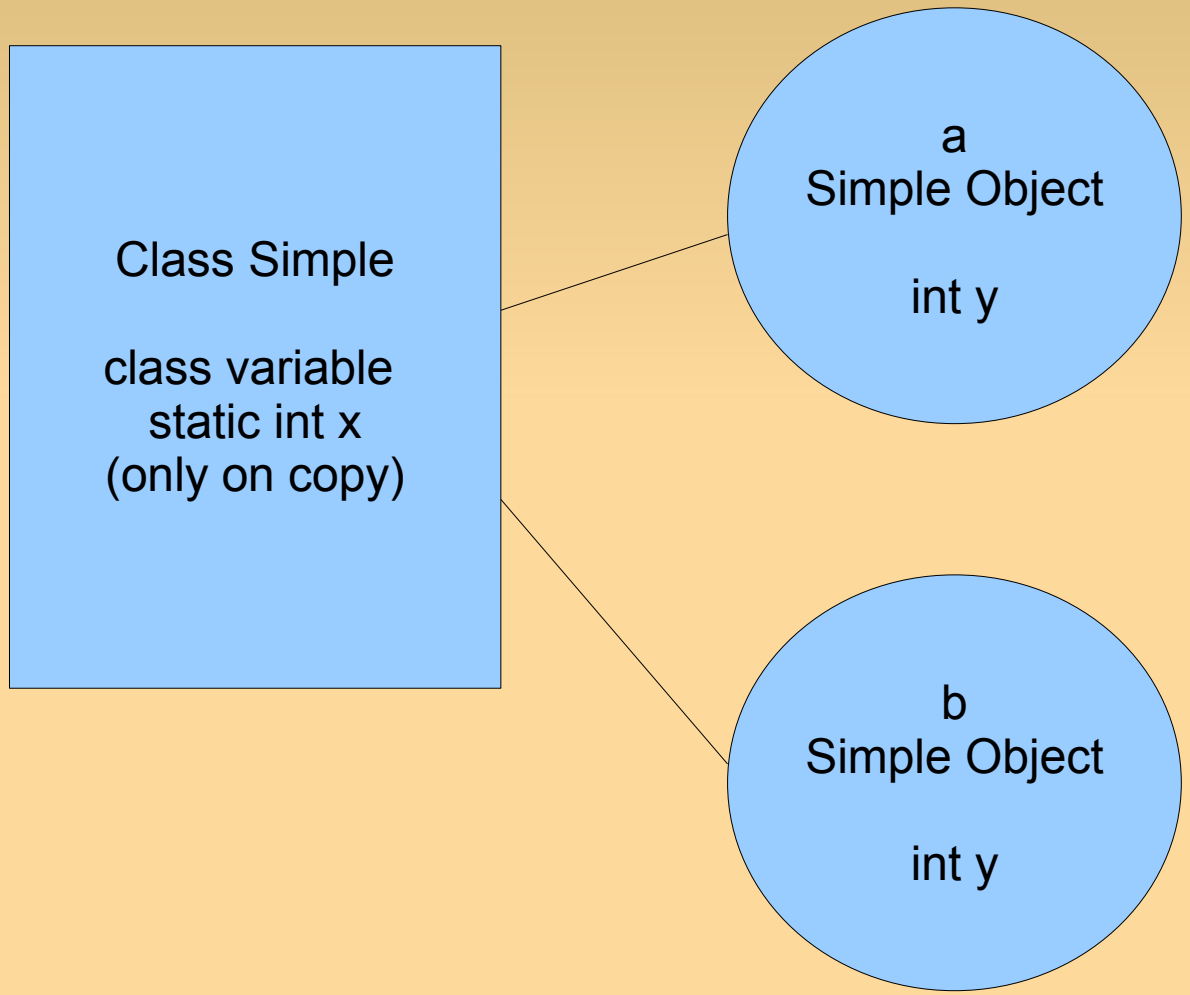
- Instance
- Instance variable
- Instance method

```
class Simple {
   static int x;
   int y;
}


class MyMain {

   public static void main (String[] args) {
      Simple a = new Simple();
      Simple b = new Simple();
   }

}
```



Instance methods

```
class Simple {
   static int x;
   int y;

   static int getX() {
      return x;
   }

   int getY() {
      return y;
   }
}


class MyMain {

   public static void main (String[] args) {
      Simple a = new Simple();
      Simple b = new Simple();
   }

}
```

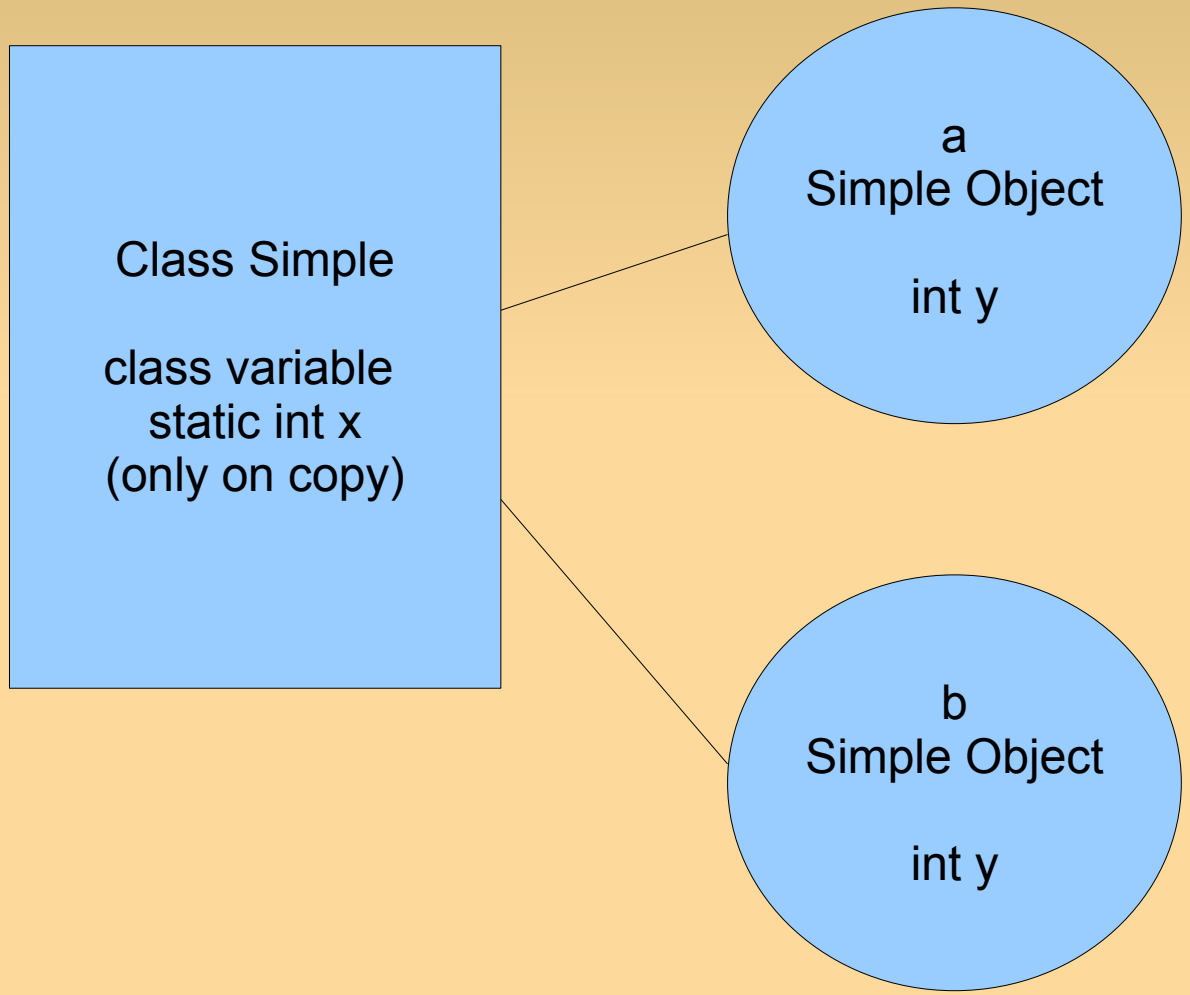


Instance methods

```
class Simple {
   static int x;
   int y;

   int getX() {
      return x;
   }

   static int getY() {
      return y;
   }
}
```

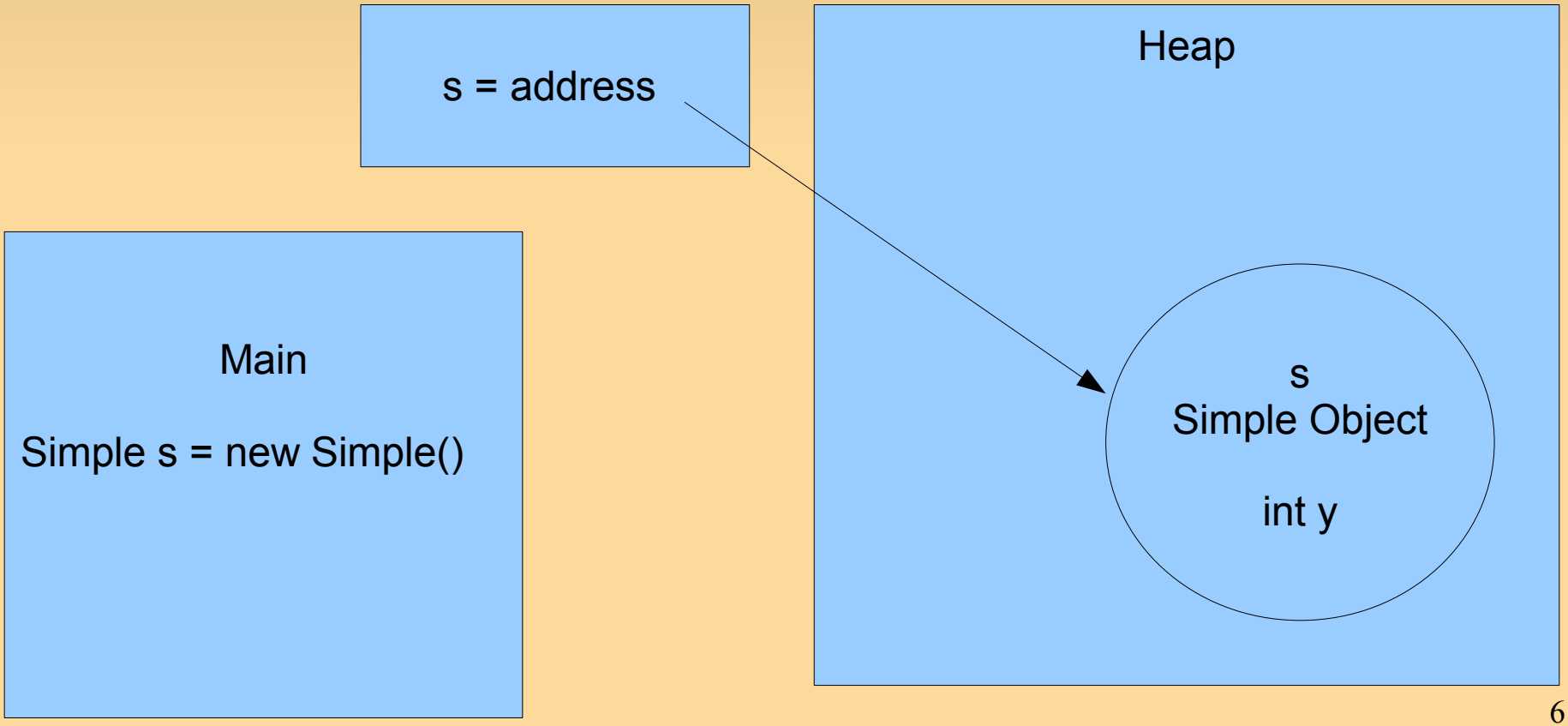
ERROR!

```
class MyMain {

   public static void main (String[] args) {
      Simple a = new Simple();
      Simple b = new Simple();
   }

}
```



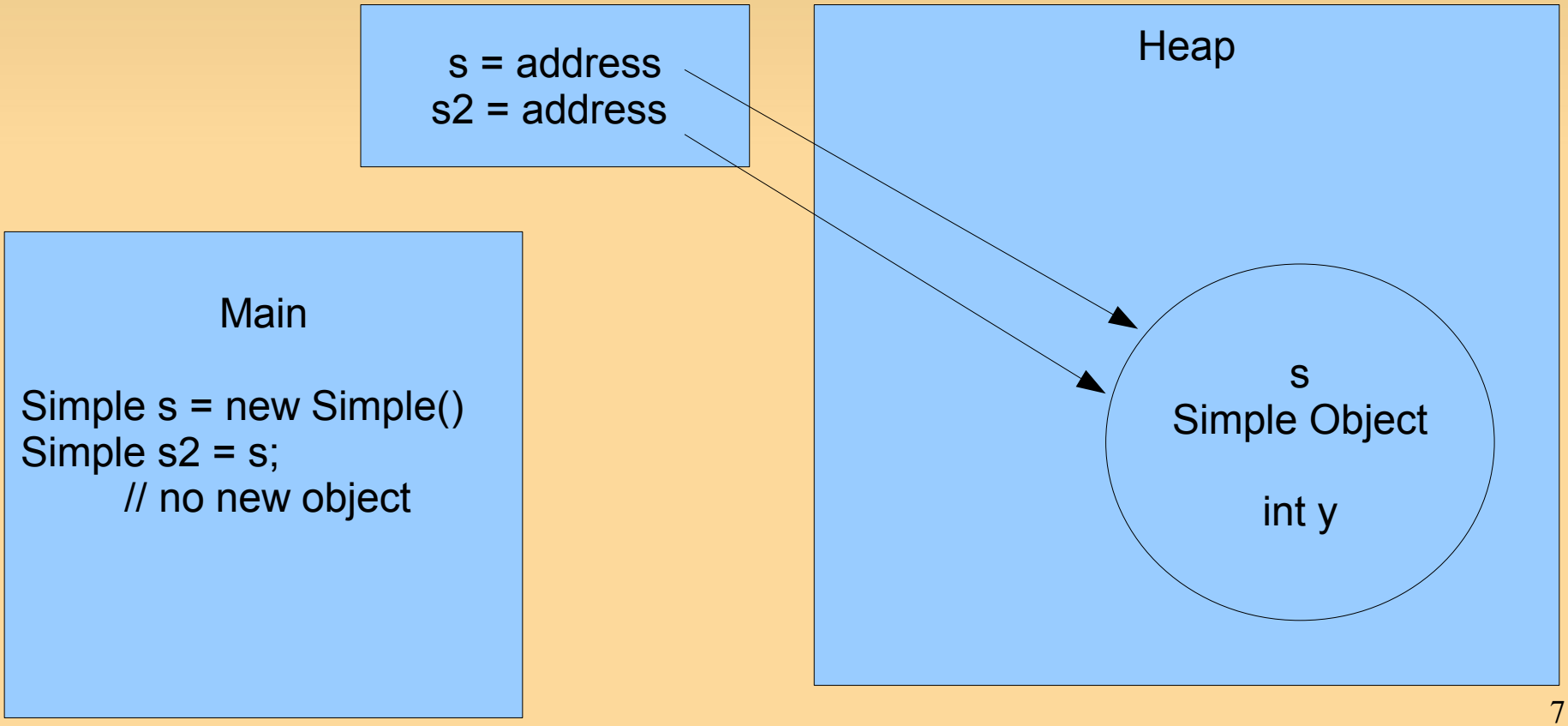
null

- References
- A variable does not contain an object



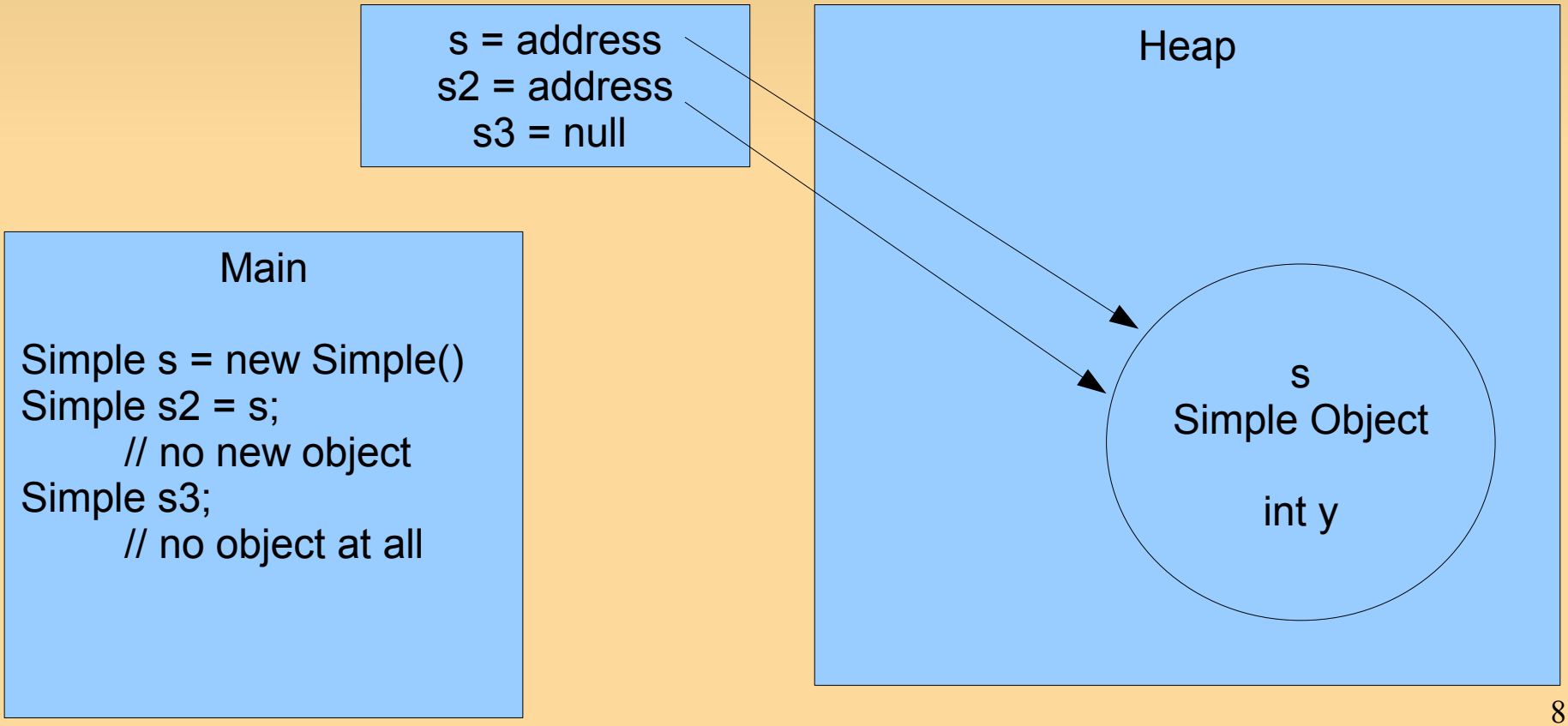
null

- References
- A variable does not contain an object



null

- References
- A variable does not contain an object



Getter and Setters

- Why not public variables
- Change all places where they have been used
-

```
class example {
   public int badIdea;
   int goodIdea;

   public void setGoodIdea(int n) throws IllegalArgumentException {
      if (n > 5) {
         throw new IllegalArgumentException("n too large");
      }
      goodIdea = n;
   }
}
```

Getter and Setters

- Why not public variables
- Change all places where they have been used

```
class example {
   public int badIdea;
   int goodIdea;
   int accessCount = 0;

   public void setGoodIdea(int n) throws IllegalArgumentException {
      if (n > 5) {
         throw new IllegalArgumentException("n too large");
      }
      goodIdea = n;
   }

   public int getGoodIdea() {
      accessCount++;
      return goodIdea;
   }
}
```

Constructors

- Object initialization
- Initializing instance variables
- Default constructor
- Same name as class
- Many constructors with different parameters

Example

```
public class PairOfDice {

    public int die1;   // Number showing on the first die.
    public int die2;   // Number showing on the second die.

    public PairOfDice() {
            // Constructor.  Rolls the dice, so that they initially
            // show some random values.
        roll();  // Call the roll() method to roll the dice.
    }

    public PairOfDice(int val1, int val2) {
            // Constructor.  Creates a pair of dice that
            // are initially showing the values val1 and val2.
        die1 = val1;  // Assign specified values
        die2 = val2;  //             to the instance variables.
    }

    public void roll() {
            // Roll the dice by setting each of the dice to be
            // a random number between 1 and 6.
        die1 = (int)(Math.random()*6) + 1;
        die2 = (int)(Math.random()*6) + 1;
    }

} // end class PairOfDice
```

```
public class RollTwoPairs {

    public static void main(String[] args) {

        PairOfDice firstDice;  // Refers to the first pair of dice.
        firstDice = new PairOfDice();

        PairOfDice secondDice; // Refers to the second pair of dice.
        secondDice = new PairOfDice();

        int countRolls;  // Counts how many times the two pairs of
                         //    dice have been rolled.

        int total1;      // Total showing on first pair of dice.
        int total2;      // Total showing on second pair of dice.

        countRolls = 0;

        do {  // Roll the two pairs of dice until totals are the same.

            firstDice.roll();    // Roll the first pair of dice.
            total1 = firstDice.die1 + firstDice.die2;   // Get total.
            System.out.println("First pair comes up  " + total1);

            secondDice.roll();    // Roll the second pair of dice.
            total2 = secondDice.die1 + secondDice.die2;   // Get total.
            System.out.println("Second pair comes up " + total2);

            countRolls++;   // Count this roll.

            System.out.println();  // Blank line.

        } while (total1 != total2);

        System.out.println("It took " + countRolls
                           + " rolls until the totals were the same.");

    } // end main()

} // end class RollTwoPairs
```

Calling a constructor

- new <class-name> (<parameter-list>)

Wrapper classes, Autoboxing

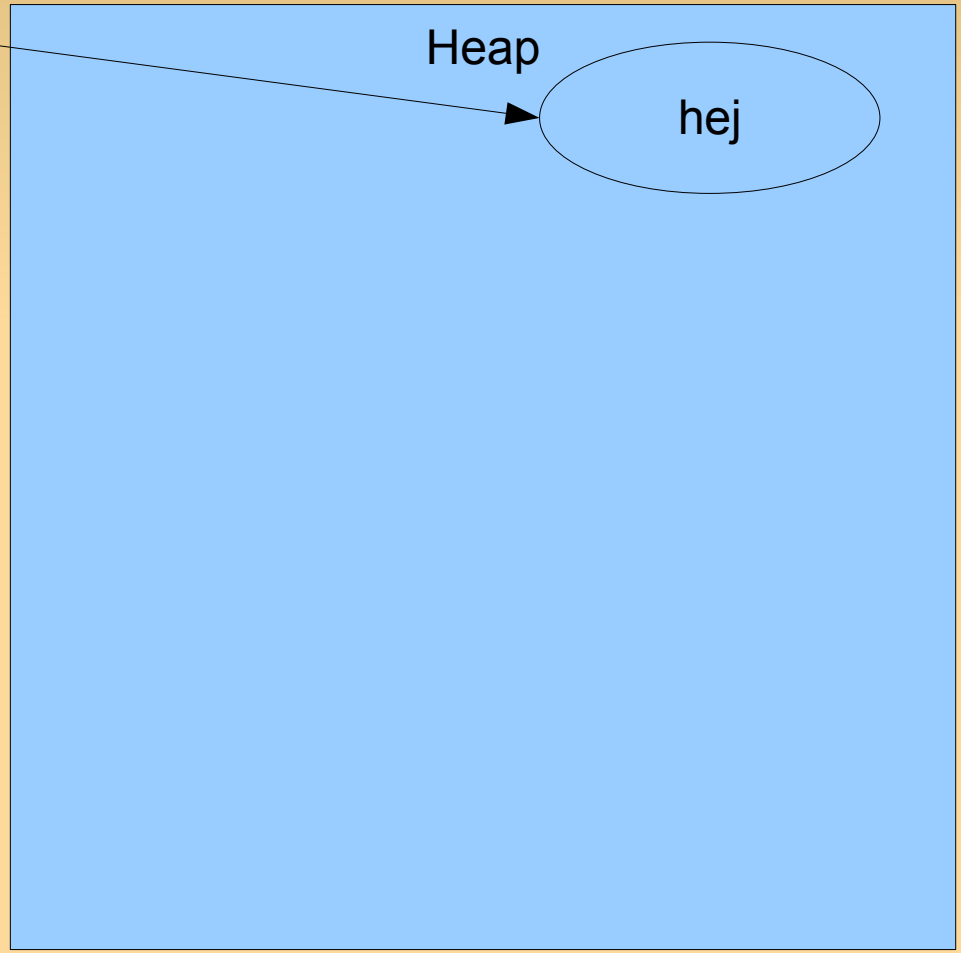
- We have seen wrapper classes
- Integer x;
- int h = 20
- x = h //Autoboxing
- x = x*20 //unboxing

Garbage collection

- Heap
- Object that no one know exist
- Manual memory management
- Leaking memory
- Garbage collection

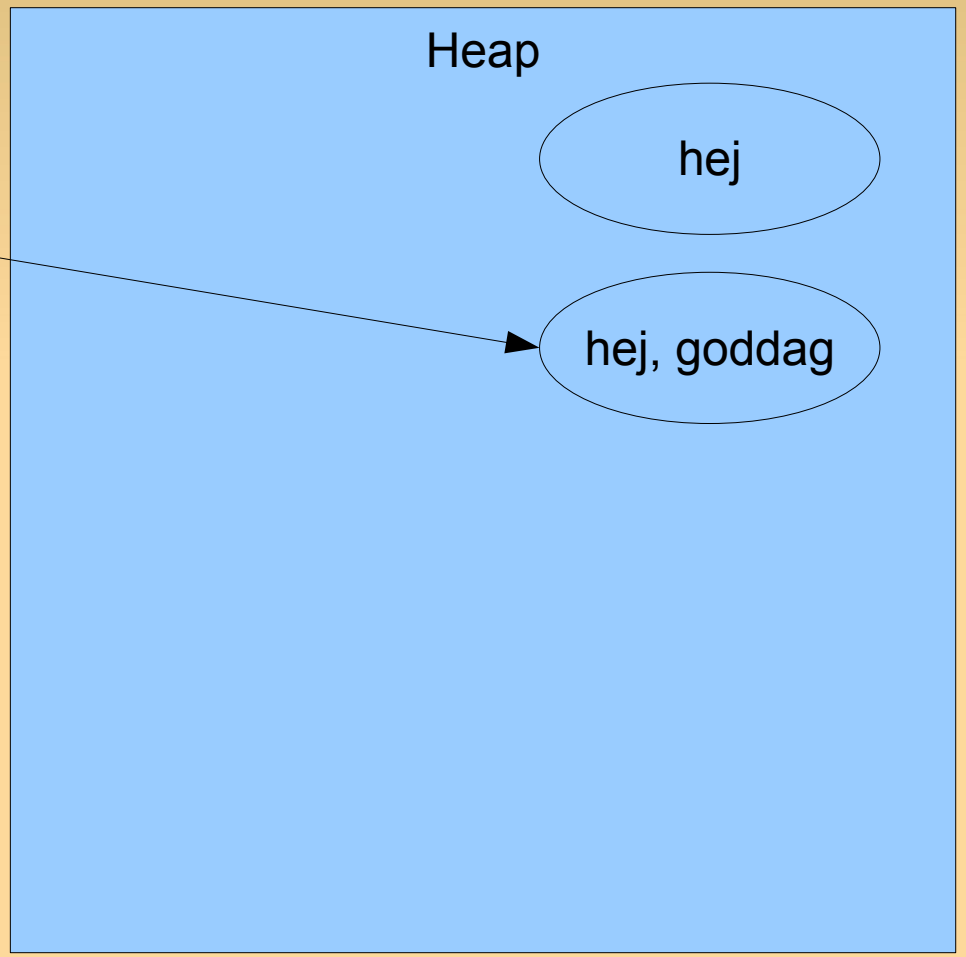
StringBuffer

- String b = “hej”
- b = b + “, goddag”
- b = b + “ og hallo”
- New object everytime
- StringBuffer b2 = “hej”
- b2.append(“, goddag”)
- b2.append(“ og hallo”)



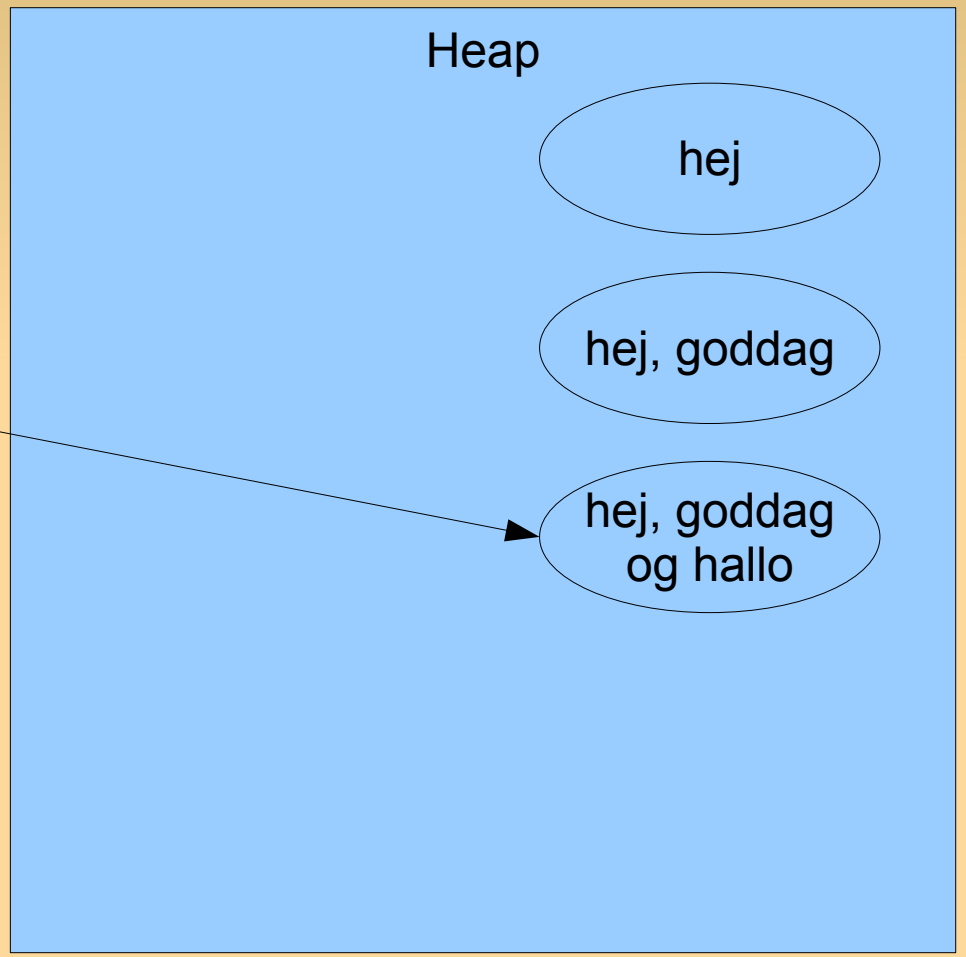
StringBuffer

- String b = “hej”
- b = b + “, goddag”
- b = b + “ og hallo”
- New object everytime
- StringBuffer b2 = “hej”
- b2.append(“, goddag”)
- b2.append(“ og hallo”)



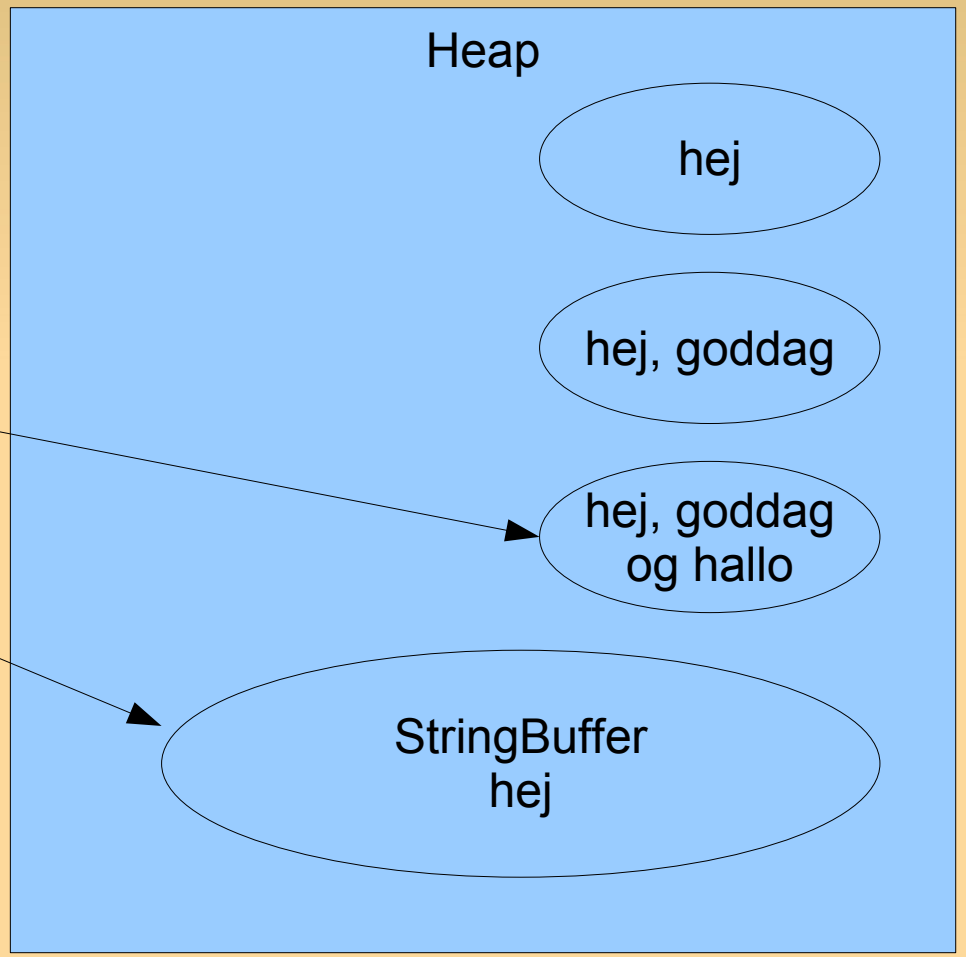
StringBuffer

- String b = “hej”
- b = b + “, goddag”
- b = b + “ og hallo”
- New object everytime
- StringBuffer b2 = “hej”
- b2.append(“, goddag”)
- b2.append(“ og hallo”)



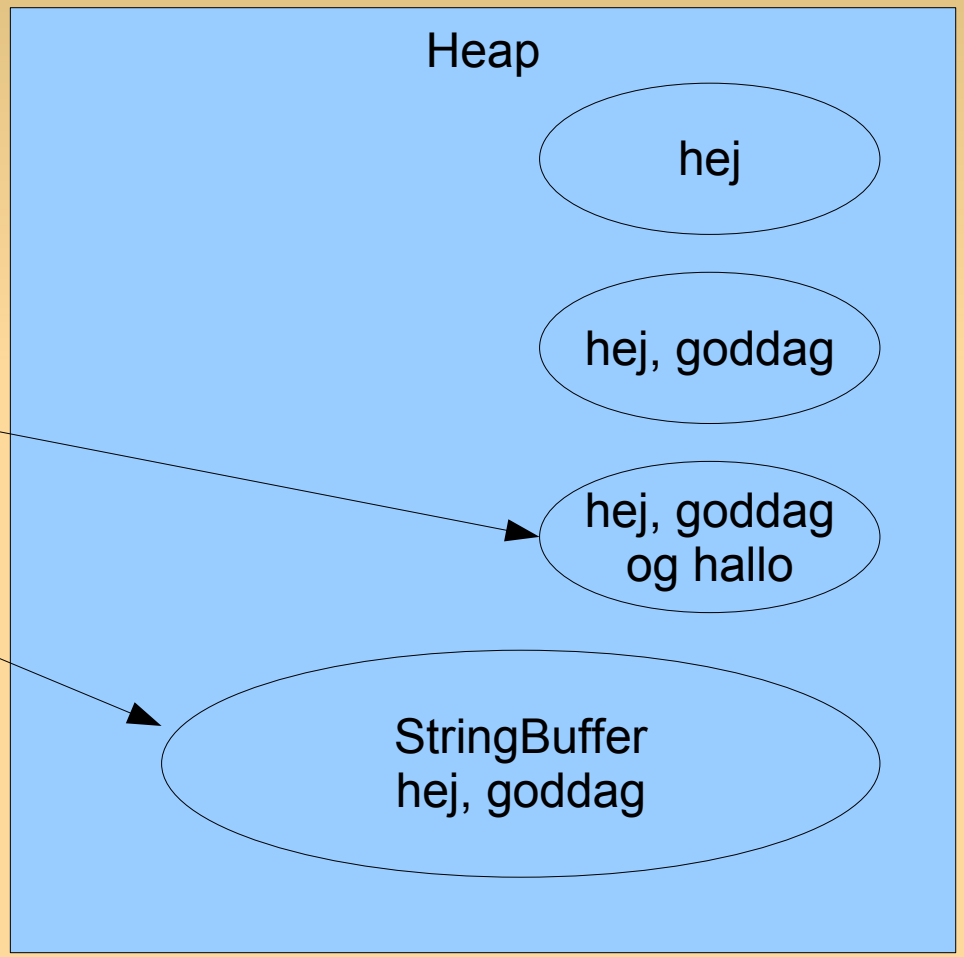
StringBuffer

- String b = “hej”
- b = b + “, goddag”
- b = b + “ og hallo”
- New object everytime
- StringBuffer b2 = “hej”
- b2.append(“, goddag”)
- b2.append(“ og hallo”)



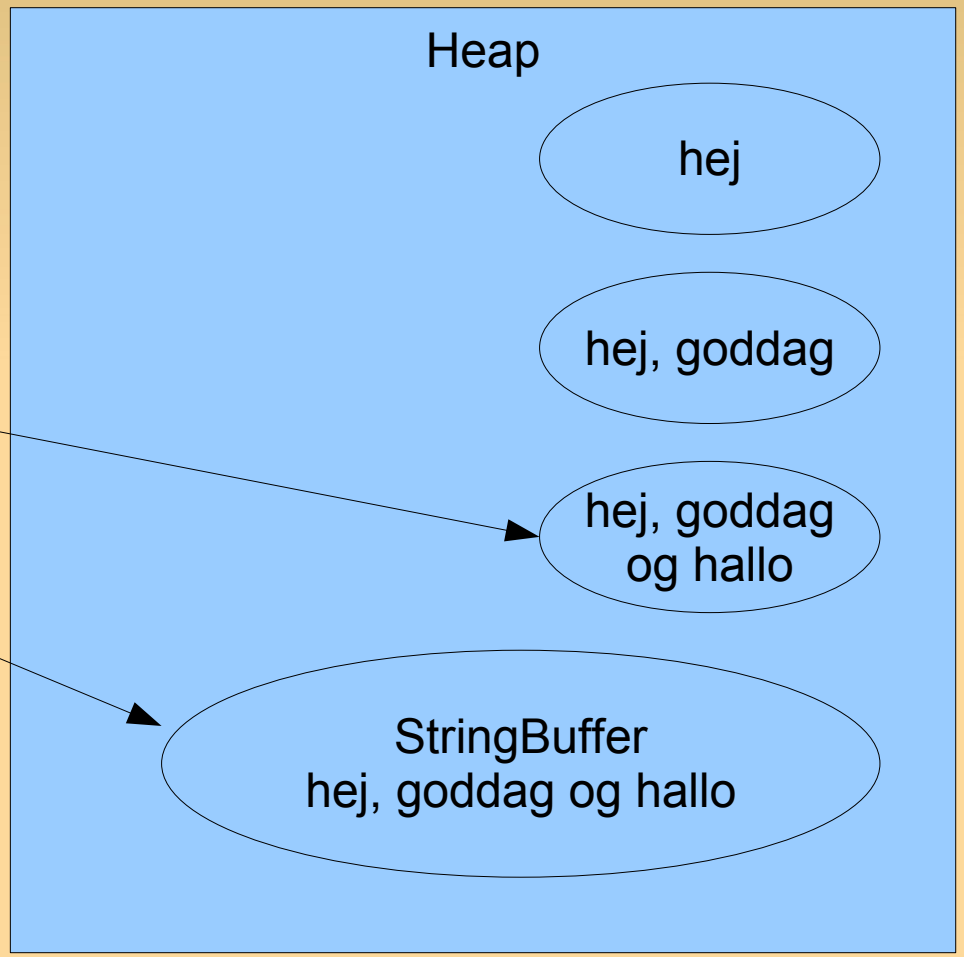
StringBuffer

- String b = “hej”
- b = b + “, goddag”
- b = b + “ og hallo”
- New object everytime
- StringBuffer b2 = “hej”
- b2.append(“, goddag”)
- b2.append(“ og hallo”)



StringBuffer

- String b = “hej”
- b = b + “, goddag”
- b = b + “ og hallo”
- New object everytime
- StringBuffer b2 = “hej”
- b2.append(“, goddag”)
- b2.append(“ og hallo”)



OOA&D

- Object Oriented Analysis and Design
- Programming with objects
- Reusable components