University at Buffalo
The State University of New York

cse@buffalo

CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce

343 Davis Hall

alphonce@buffalo.edu

Office hours:

Tuesday 10:00 AM – 12:00 PM*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

OR request appointment via e-mail

Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6

ANNOUNCEMENTS

EXAM 1 REMINDER & ROOMS

Tuesday October 4

8:45 PM – 9:45 PM

LOCATION: tentative assignments – subject to change

LAST NAME	**ROOM**
Aarabi – Desso	*NSC 201*
Dhakite – Fung	*NSC 216*
Gagne – Higgins	*NSC 220*
Homeyer – Pinkhasik	*NSC 225*
Prais – Villegas	*Knox 104*
Wager – Zykov	*Davis 101*

BRING: your UB card

NO ELECTRONICS: cell phone, calculator, etc.

EXAM 1 EXTRA REVIEW SESSIONS:

Come prepared with questions!

Sat Oct 1 2016 4:00PM - 5:30PM in Davis 101

Mon Oct 3 2016 5:00PM - 6:20PM in Knox 110

(Monday's lecture will also be a review/Q&A session)

ELECTRONICS:

off & away

ROADMAP

Last time

- Relationships (continued)
  - association
  - accessor/mutator methods

Today

- accessor/mutator methods
- modeling

Coming up

- Relationships (continued)

REVIEW

Mutator method: setting a variable's value

A method which changes the value of an instance variable.

Allows us to grant WRITE access to the contents of a variable which itself is PRIVATE.

Association: constructor & mutator method

```
public class Dog {

    private Collar _collar;

    public Dog(Collar c) {
        _collar = c;
    }

    public void setCollar(Collar c) {
        _collar = c;
    }

}
```

Accessor method: getting a variable's value

A method which returns the value of an instance variable

Allows us to grant READ access to the contents of a variable which itself is PRIVATE.

Accessor method

```
public class Dog {

  private Collar _collar;

  public Dog(Collar c){
    _collar = c;
  }

  public Collar getCollar(){
    return _collar;
  }
}
```

MOVING ON

MODELING

(also: execution model)

EXERCISE

The class definition

```
public class Shape {
   private java.awt.Color _color;
   public Shape(java.awt.Color c) {
      _color = c;
   }
   public java.awt.Color getColor() {
      return _color;
   }
   public void setColor(java.awt.Color c){
      _color = c;
   }
}
```

static variables / java.awt.Color

```
Shape s1 = new Shape(java.awt.Color.BLUE);
Shape s2 = new Shape(java.awt.Color.RED);
s2.setColor(s1.getColor());
```

What are the highlighted expressions?

Exercise

```
Shape s1 = new Shape(java.awt.Color.BLUE);
Shape s2 = new Shape(java.awt.Color.RED);
s2.setColor(s1.getColor());
```

Discuss with your neighbors what happens.

What will the object diagram look like after the method calls happen?

University at Buffalo
The State University of New York

cse@buffalo

What we did in class…

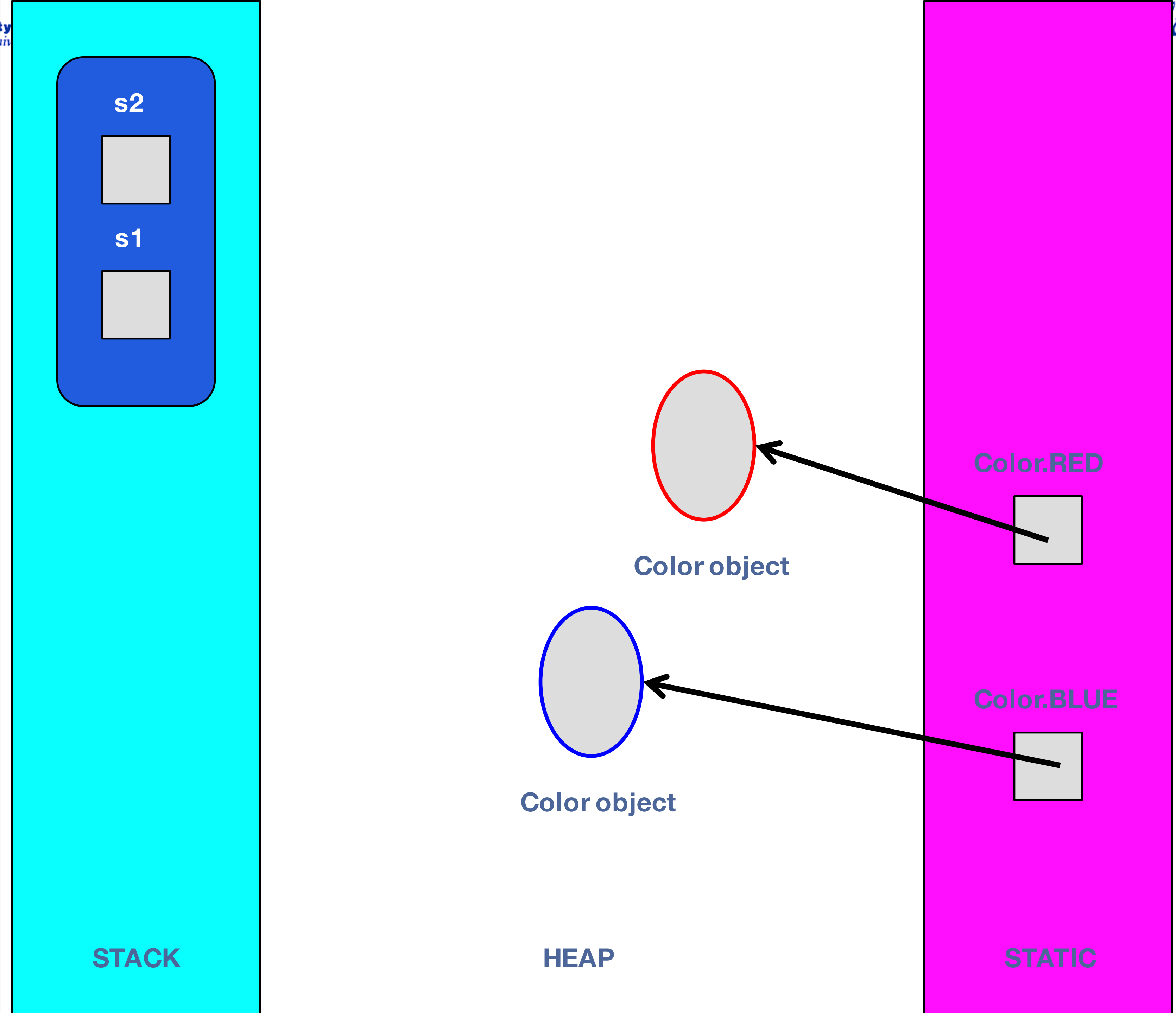
We considered this example:

```
Shape s1 = new Shape(java.awt.Color.BLUE);
Shape s2 = new Shape(java.awt.Color.RED);
s2.setColor(s1.getColor());
```

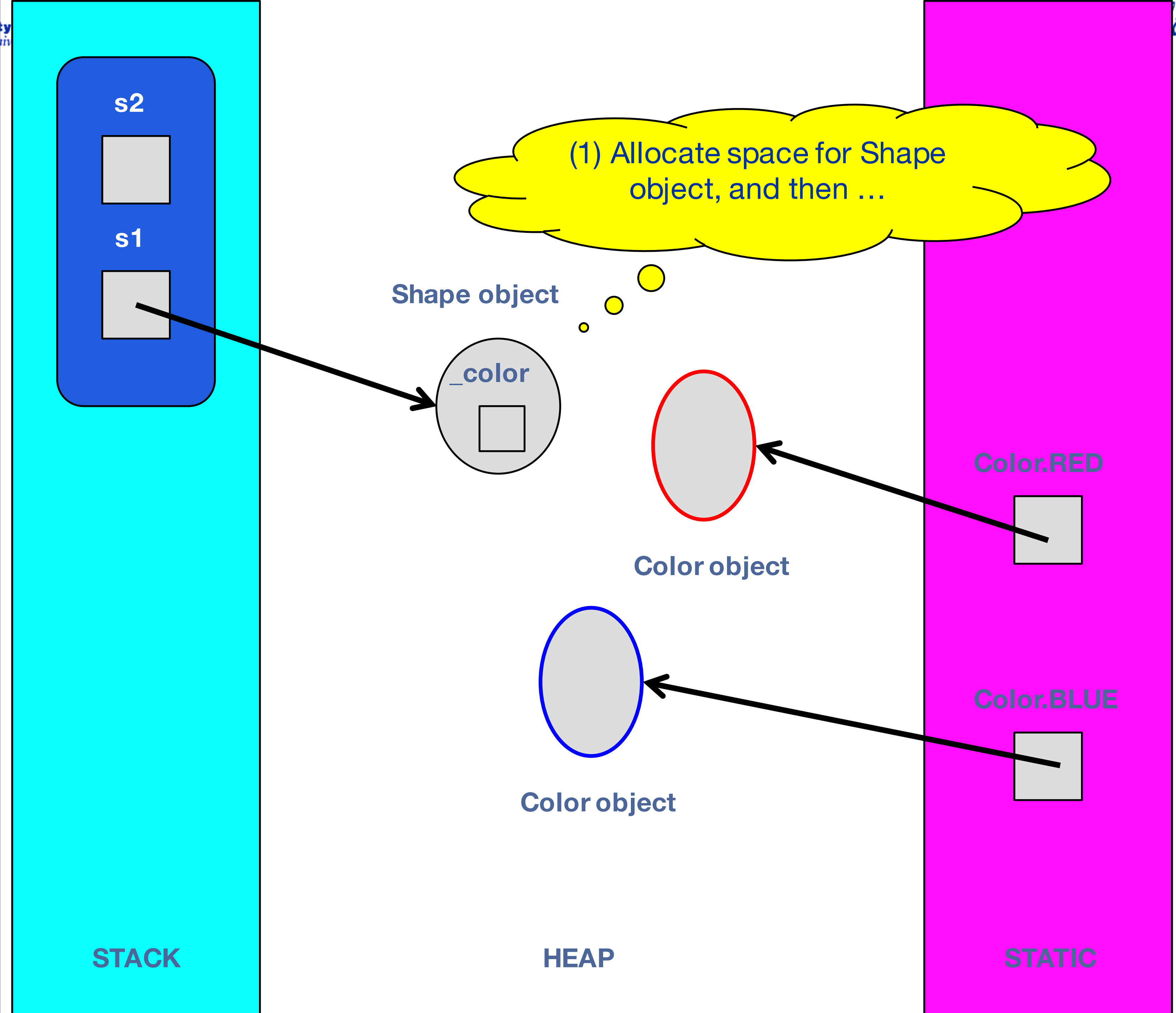
Let's work through this in detail.
What happens during the execution of:

```
    s1 = new Shape(java.awt.Color.BLUE)
```

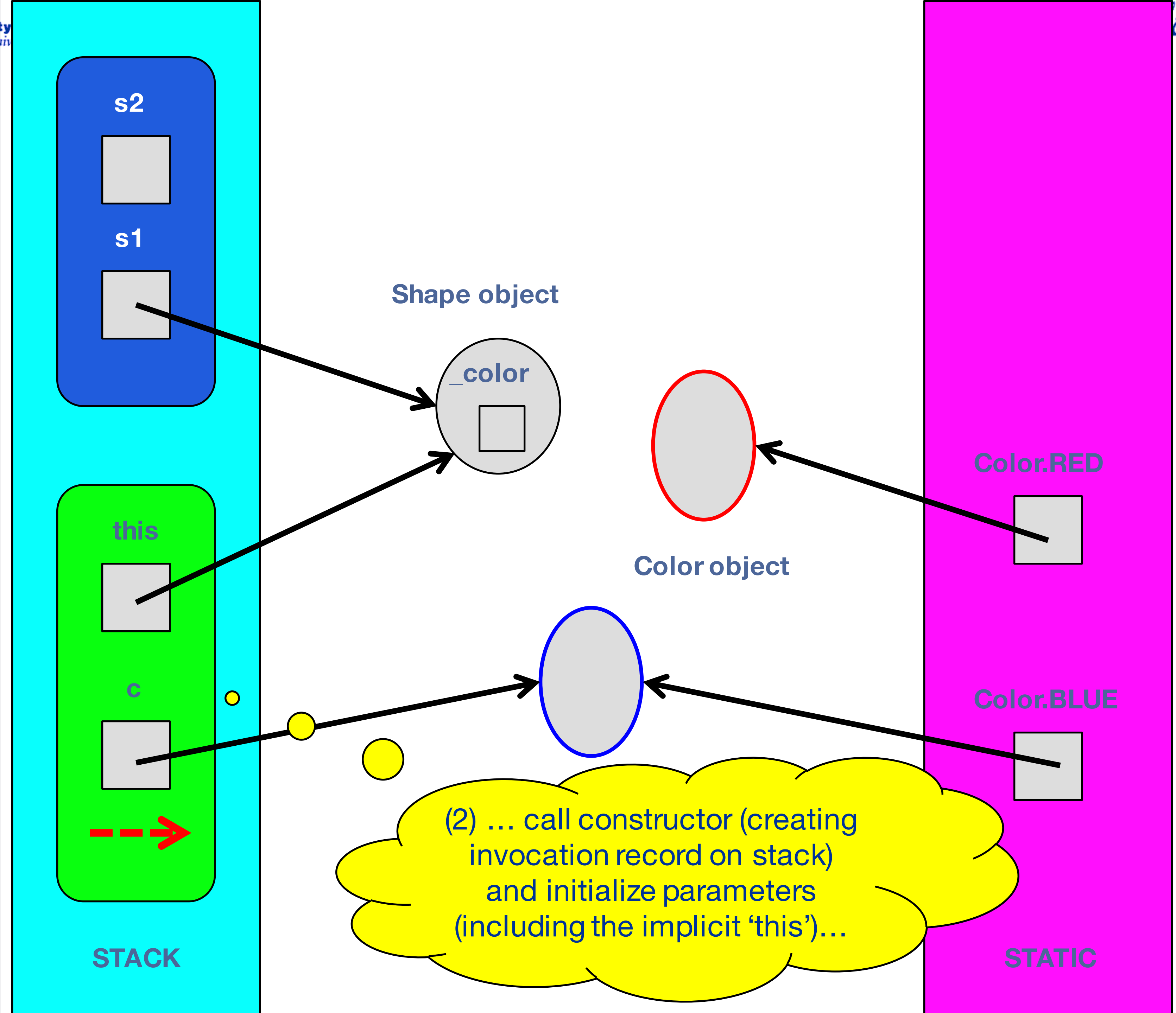
s1 = new Shape(java.awt.Color.BLUE)



```
s1 = new Shape(java.awt.Color.BLUE)
```

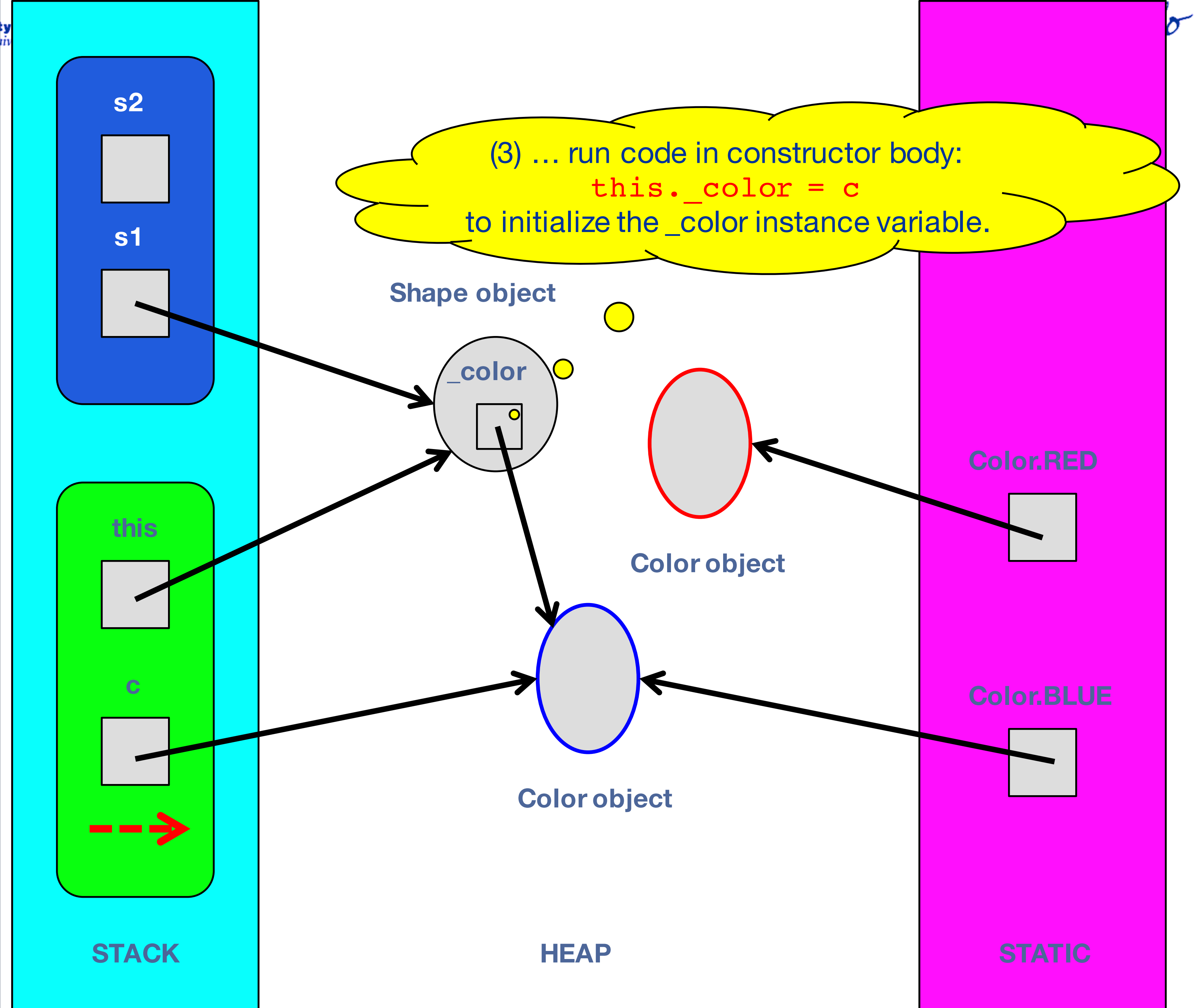


```
s1 = new Shape(java.awt.Color.BLUE)
```

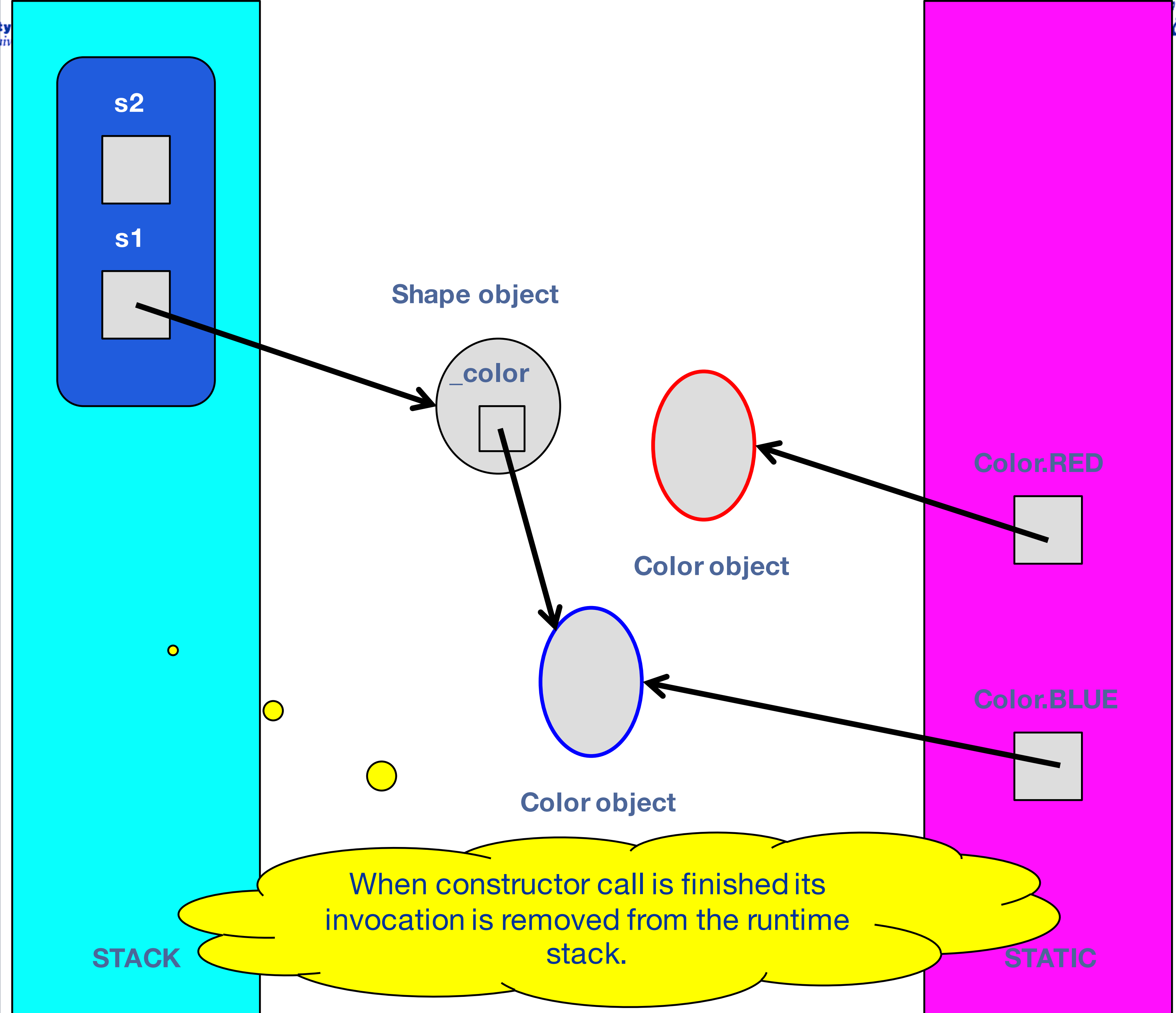


(2) … call constructor (creating invocation record on stack) and initialize parameters (including the implicit 'this')…

s1 = new Shape(java.awt.Color.BLUE)



s1 = new Shape(java.awt.Color.BLUE)



When constructor call is finished its invocation is removed from the runtime stack.

The class definition

```
public class Shape {
   private java.awt.Color _color;
   public Shape(java.awt.Color c) {
      _color = c;
   }
   public java.awt.Color getColor() {
      return _color;
   }
   public void setColor(java.awt.Color c){
      _color = c;
   }
}
```

Exercise

```
Shape s1 = new Shape(java.awt.Color.BLUE);
Shape s2 = new Shape(java.awt.Color.RED);
s2.setColor(s1.getColor());
```

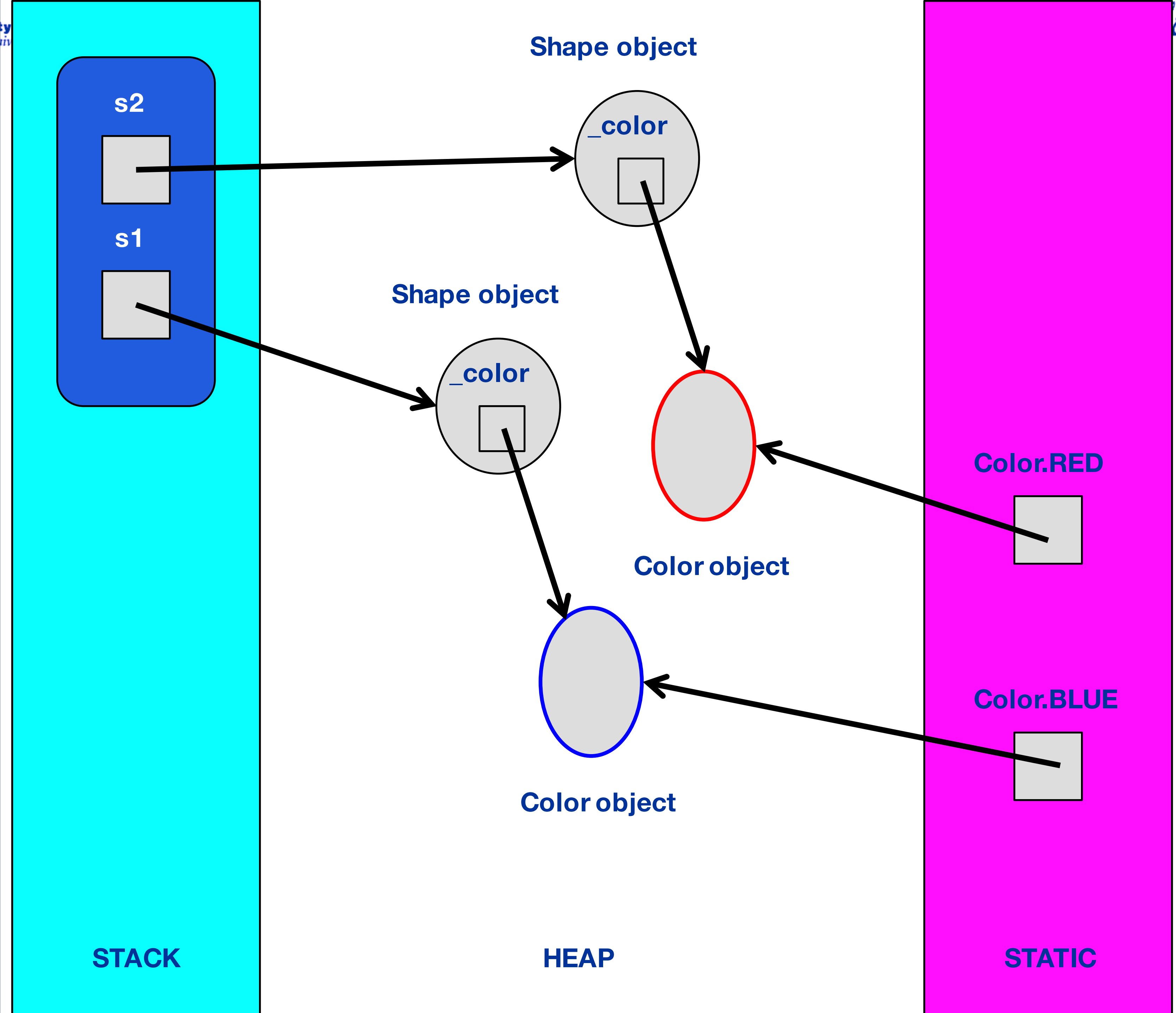
What will the object diagram look like after the method calls happen?

WE WILL NOW GO THROUGH, STEP BY STEP, WHAT HAPPENS.

IF YOU HAVE QUESTIONS, ASK THEM!!

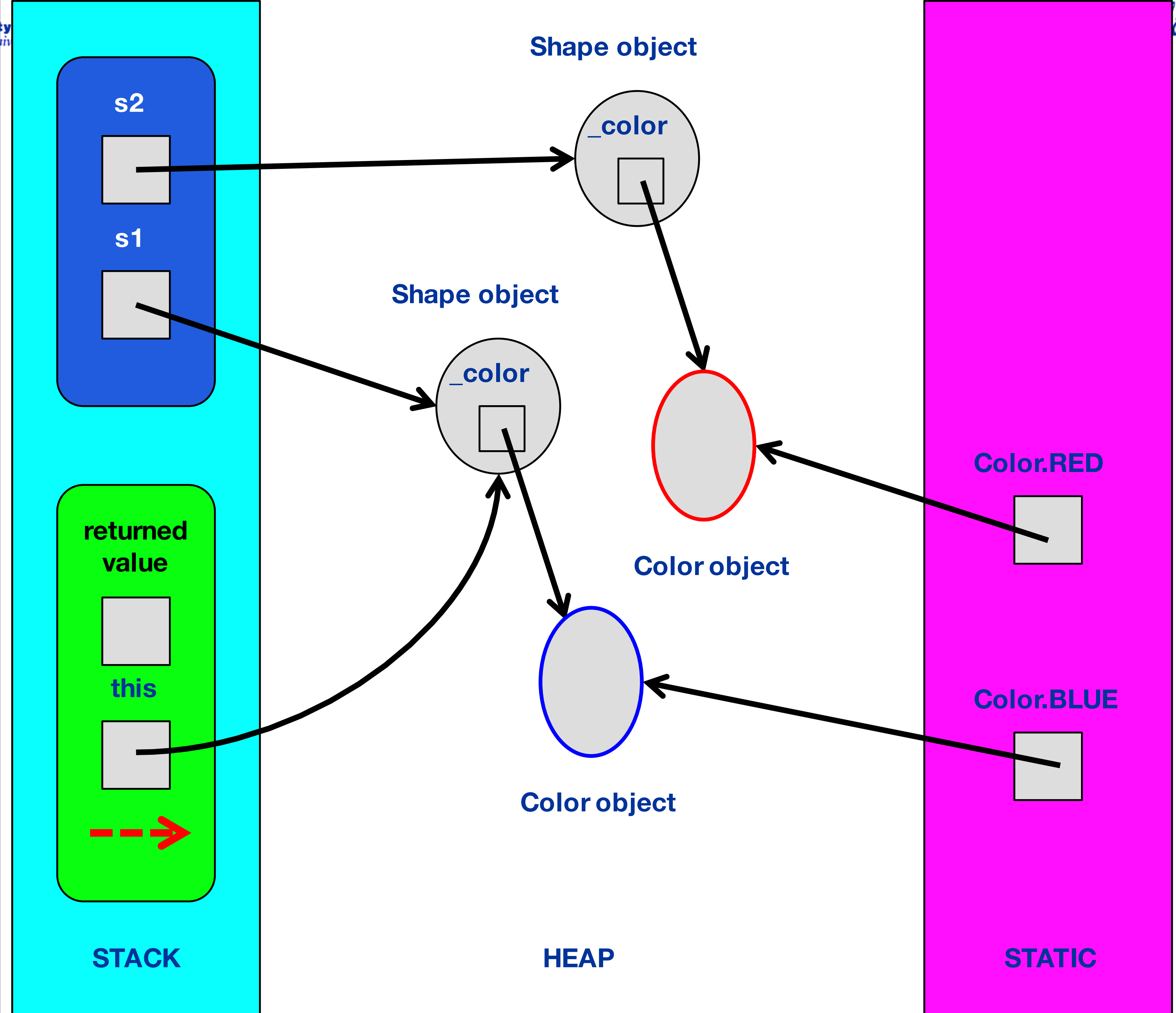
Before call:

`s2.setColor(s1.getColor())`



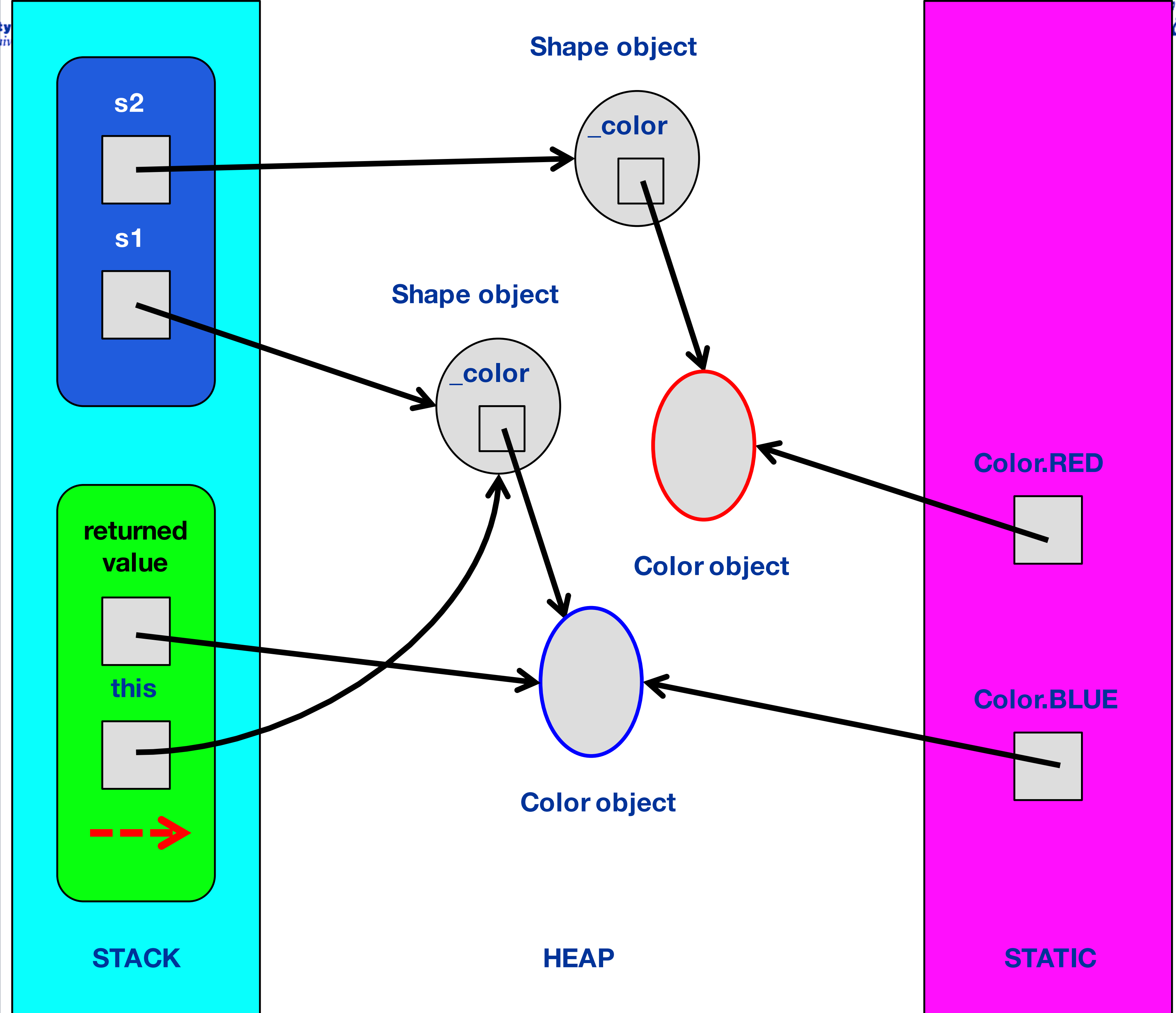
At call entry:

```
s2.setColor(s1.getColor())
```



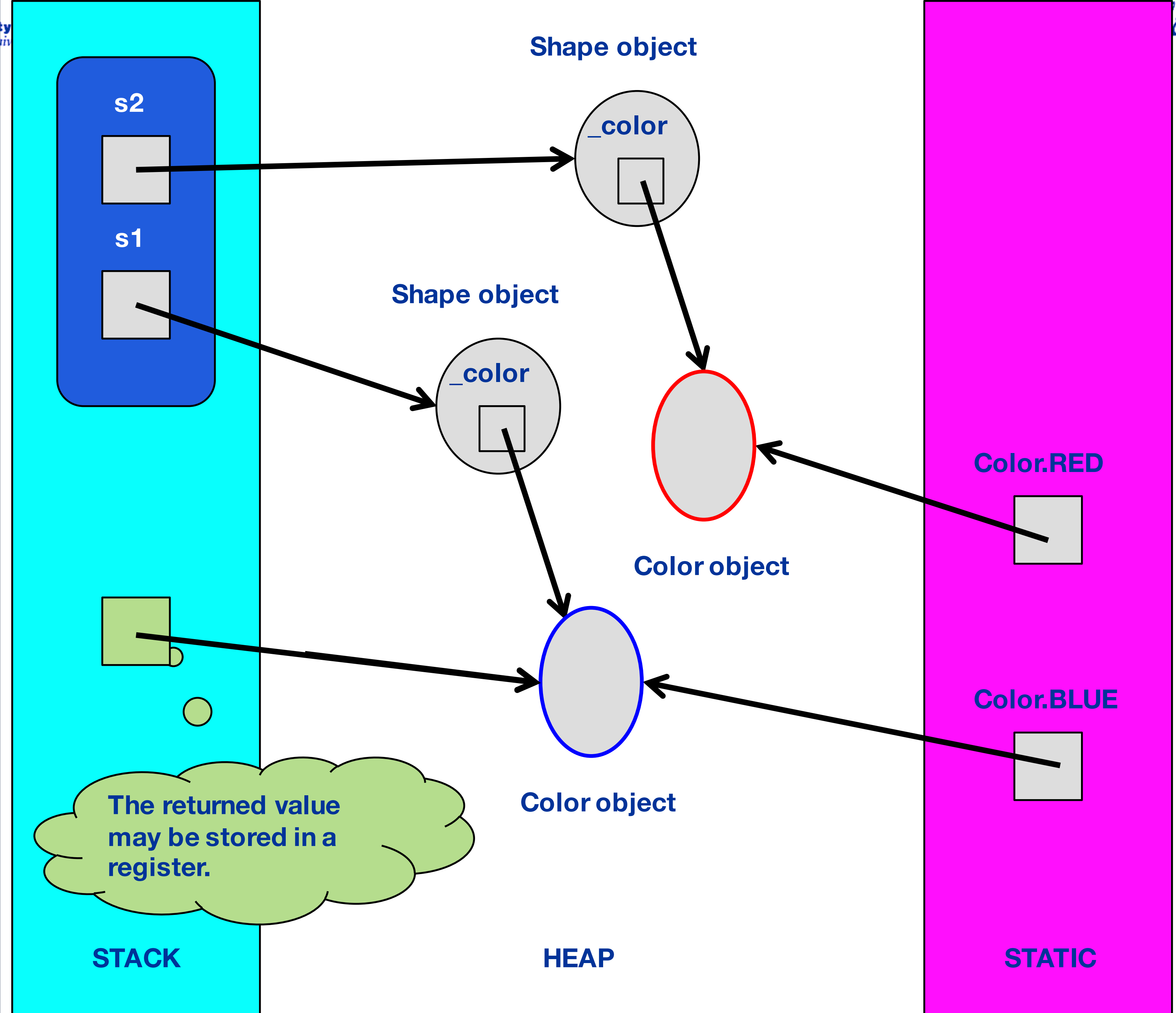
At call exit:

`s2.setColor(s1.getColor())`

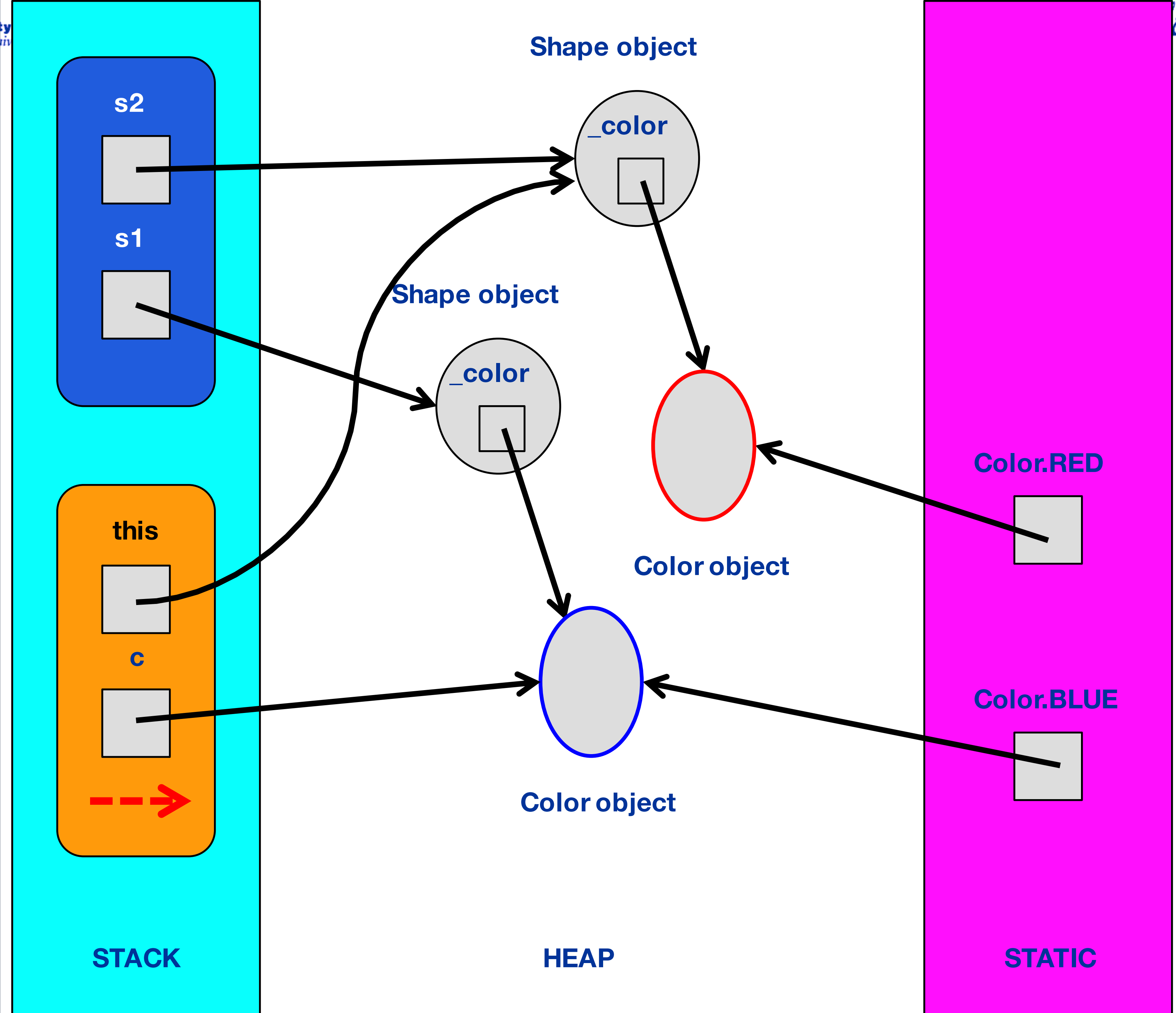


The returned value is held in a temporary:

`s2.setColor(s1.getColor())`

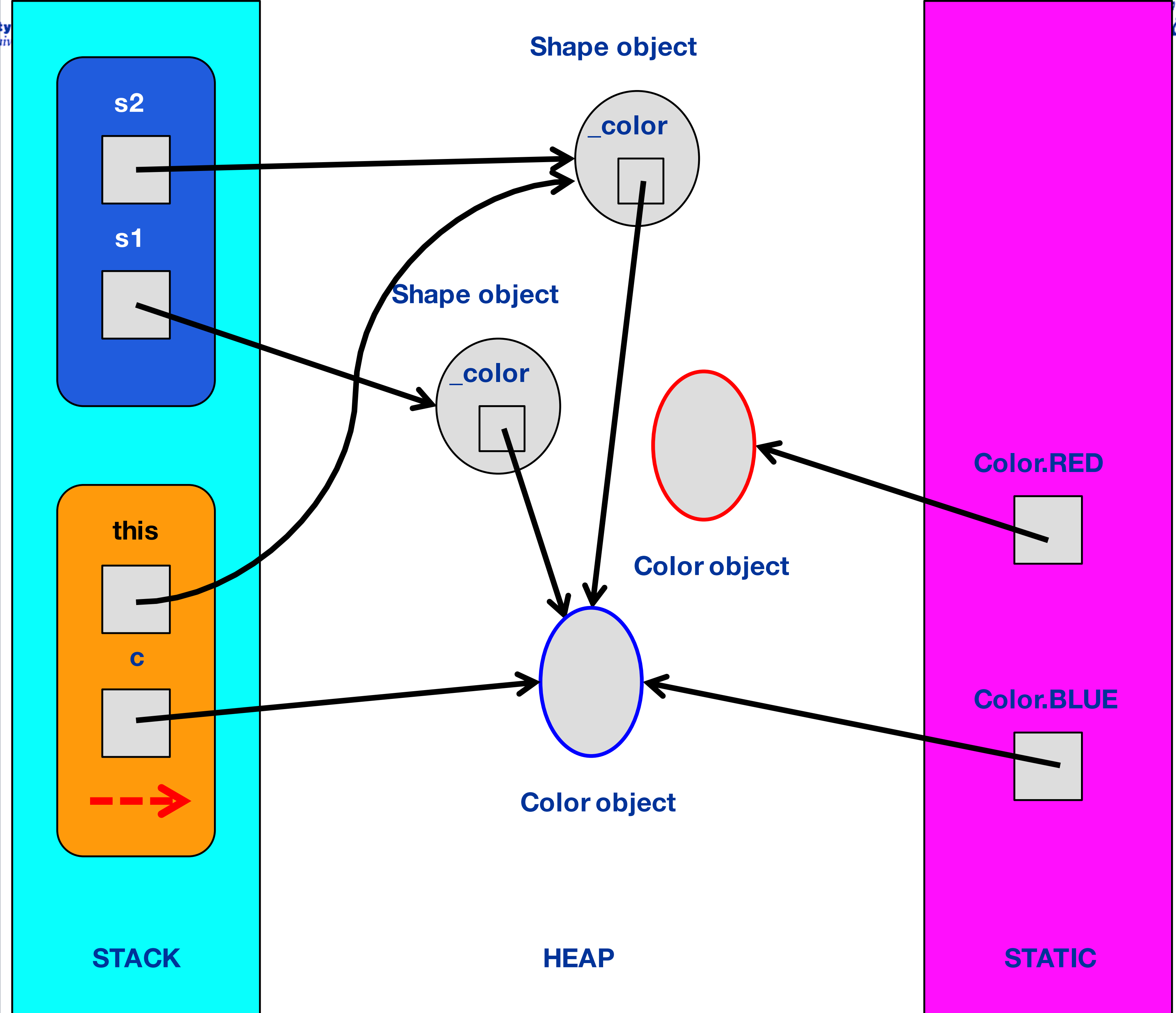


At call entry:
`s2.setColor(s1.getColor())`

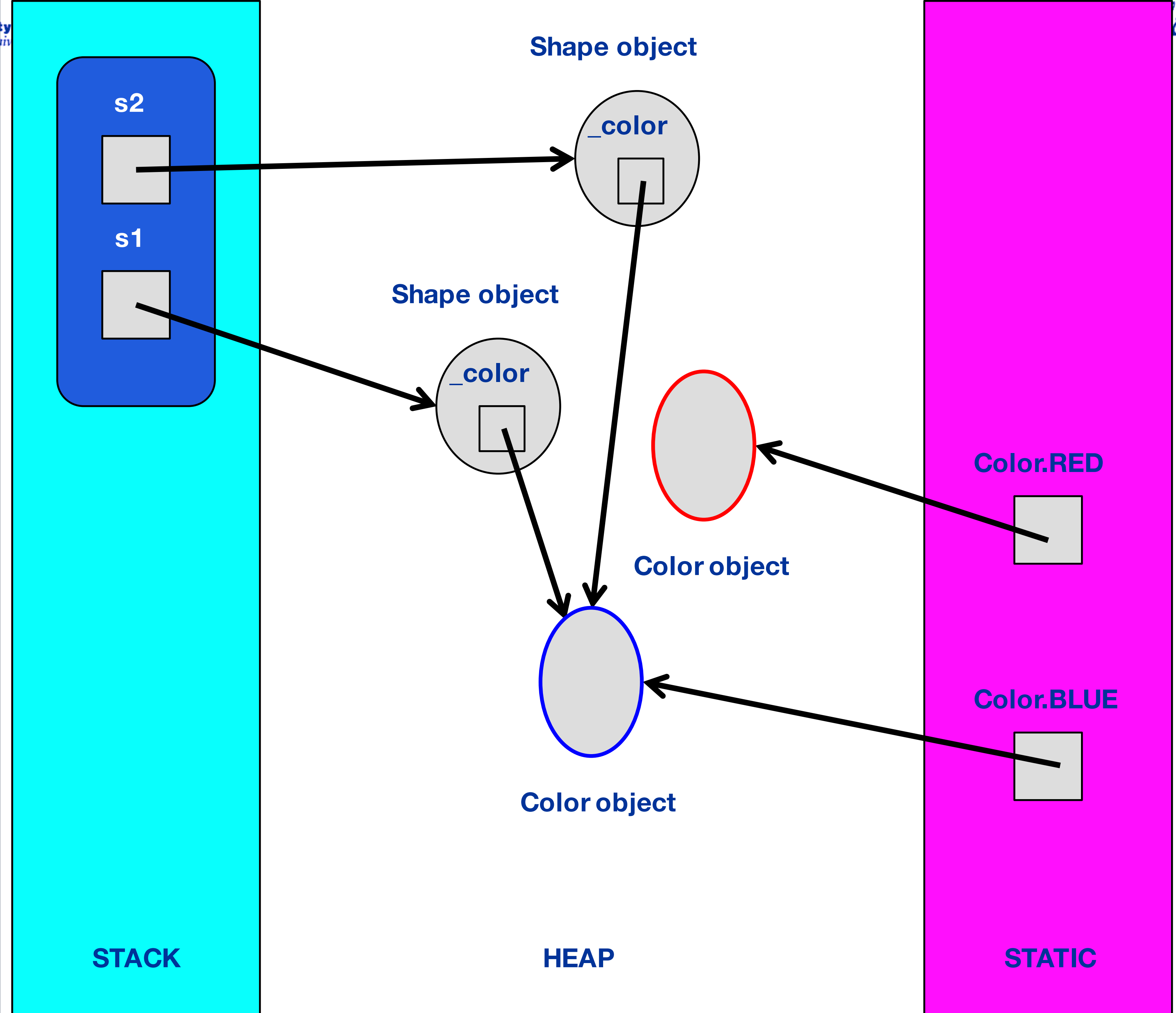


At call exit:

`s2.setColor(s1.getColor())`



After call exit: `s2.setColor(s1.getColor())`



Result?

Both shapes have the same color (java.awt.Color.BLUE).

This is OK.