Test 1

- When: this week, during your lab
- What: Chapter 2, excluding Section 2.6 (textbook will be available)
- Type of questions: one programming question and one multiple choice question
- Textbook: study it, since studying just the slides might not be enough
- Lab: attend the lab in which you are officially enrolled so that we can ensure that there is a computer for everyone

Implementing non-static features

Problem

Implement the Rectangle class.

What have we done so far?

- attributes
- constructors
- accessors
- mutators
- getArea
- toString
- scale

Still to do?

- equals
- compareTo
- hashCode

equals method

Question

What is the difference between the `equals` method and the == operator?

equals method

Question

What is the difference between the `equals` method and the == operator?

Answer

The `equals` method checks whether two objects have the same state, whereas the == operator checks if two objects have the same identity.

equals method

Let have a look at the API of the `equals` method of the `Object` class.

Fact

The `equals` method of every class has to satisfy the properties specified in the API of the `equals` method of the `Object` class. Later, we will discuss the reason why.

Equivalence relation

Definition

Let X be a set. A relation $R \subseteq X \times X$ is an equivalence relation if for all x, y, $z \in X$,

- $(x, x) \in R$
  (reflexivity)
- if $(x, y) \in R$ then $(y, x) \in R$
  (symmetry)
- if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$
  (transitivity)

Equivalence relation

Example

Let S be the set of students in this course. Two students s_1 and s_2 are in the relation B if they have the same birthday. The relation B is an equivalence relation, because for all students s_1, s_2 and s_3,

- s_1 has the same birthday as s_1
  $(s_1, s_1) \in B$
- if s_1 has the same birthday as s_2 then s_2 has the same birthday as s_1
  if $(s_1, s_2) \in B$ then $(s_2, s_1) \in B$
- if s_1 has the same birthday as s_2 and s_2 has the same birthday as s_3 then s_1 has the same birthday as s_3
  if $(s_1, s_2) \in B$ and $(s_2, s_3) \in B$ then $(s_1, s_3) \in B$

Equivalence relation

Example

Let $\mathbb{Z}$ be the set of integers. Two integers x and y are in the relation S if $x^2 = y^2$. The relation S is an equivalence relation, because for all integers x, y and z,

- $x^2 = x^2$
  $(x, x) \in S$
- if $x^2 = y^2$ then $y^2 = x^2$
  if $(x, y) \in S$ then $(y, x) \in S$
- if $x^2 = y^2$ and $y^2 = z^2$ then $x^2 = z^2$
  if $(x, y) \in S$ and $(y, z) \in S$ then $(x, y) \in S$

equals of Rectangle

Question

Where can find when two rectangles are considered to be the same?

equals of Rectangle

Question

Where can find when two rectangles are considered to be the same?

Answer

In the API of the `Rectangle` class.

equals of Rectangle

Two rectangles are considered the same if they have the same width and height.

Question

Let R be the set of all rectangles. Two rectangles are in the relation S if they have the same width and height. Is S an equivalence relation?

equals of Rectangle

Question

Let R be the set of all rectangles. Two rectangles are in the relation S if they have the same width and height. Is S an equivalence relation?

equals of Rectangle

Question

Let R be the set of all rectangles. Two rectangles are in the relation S if they have the same width and height. Is S an equivalence relation?

Answer

Yes, because for all rectangles r_1, r_2 and r_3,

- r_1 has the same width and height as r_1,
- if r_1 has the same width and height as r_2 then r_2 has the same width and height as r_1, and
- if r_1 has the same width and height as r_2 and r_2 has the same width and height as r_3 then r_1 has the same width and height as r_3.

equals method

Whatever the definition of equality is, the `equals` method must satisfy the following properties:

- `x.equals(x)` returns true for any `x` different from null,
- `x.equals(y)` returns true if and only if `y.equals(x)` returns true for all `x` and `y` different from null,
- if `x.equals(y)` returns true and `y.equals(z)` returns true then `x.equals(z)` returns true for all `x`, `y` and `z` different from null,
- `x.equals(null)` returns `false` for all `x` different from null.

equals of Rectangle

```
public boolean equals(Object object)
{
    boolean equal;
    ...
    return equal;
}
```

equals of Rectangle

Question

What should `equals` return if `object` is null?

equals of Rectangle

Question

What should `equals` return if `object` is null?

Answer

`false`

equals of Rectangle

Question

What should `equals` return if `object` is null?

Answer

`false`

Question

How do we capture this in the body of the `equals` method?

equals of Rectangle

Question

What should `equals` return if `object` is null?

Answer

`false`

Question

How do we capture this in the body of the `equals` method?

Answer

```
if (object == null)
{
   equal = false;
}
```

equals of Rectangle

Question

What should `equals` return if `object` is not a `Rectangle`?

equals of Rectangle

Question

What should `equals` return if `object` is not a `Rectangle`?

Answer

`false`

equals of Rectangle

Question

How can we check whether `object` is a `Rectangle`?

equals of Rectangle

Question

How can we check whether `object` is a `Rectangle`?

Answer

By means of the `instanceof` operator or the `getClass` method.

equals of Rectangle

Question

How can we check whether `object` is a `Rectangle`?

Answer

By means of the `instanceof` operator or the `getClass` method.

Question

Which is more appropriate?

equals of Rectangle

Question

How can we check whether `object` is a `Rectangle`?

Answer

By means of the `instanceof` operator or the `getClass` method.

Question

Which is more appropriate?

Answer

The `getClass` method. We will discuss later why.

getClass of Object

An object of the class `Class`, which is part of the package java.lang, represents a class. For each class, there is a unique object of the class `Class` that represents it. This unique object is returned by the method `getClass`.

getClass of Object

Question

Given two object, `first` and `second` both of type `Object`, how do you check if they are instances of the same class?

getClass of Object

Question

Given two object, `first` and `second` both of type `Object`, how do you check if they are instances of the same class?

Answer

```
if (first.getClass() == second.getClass()) { ... }
```

getClass of Object

Question

Given two object, `first` and `second` both of type `Object`, how do you check if they are instances of the same class?

Answer

```
if (first.getClass() == second.getClass()) { ... }
```

Question

Although `first.getClass()` and `second.getClass()` are objects, why can we compare them using the == operator?

getClass of Object

Question

Given two object, `first` and `second` both of type `Object`, how do you check if they are instances of the same class?

Answer

```
if (first.getClass() == second.getClass()) { ... }
```

Question

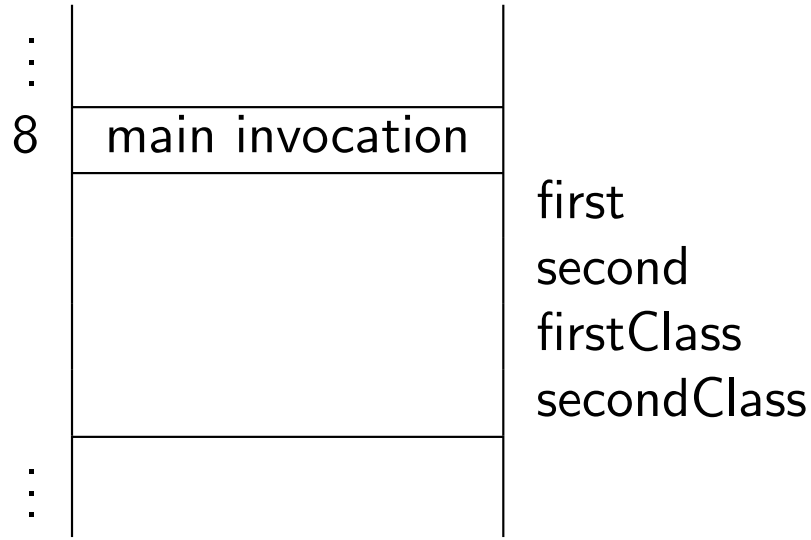
Although `first.getClass()` and `second.getClass()` are objects, why can we compare them using the == operator?

Answer

Since there is a unique `Class` object representing each class, it suffices to compare identities.

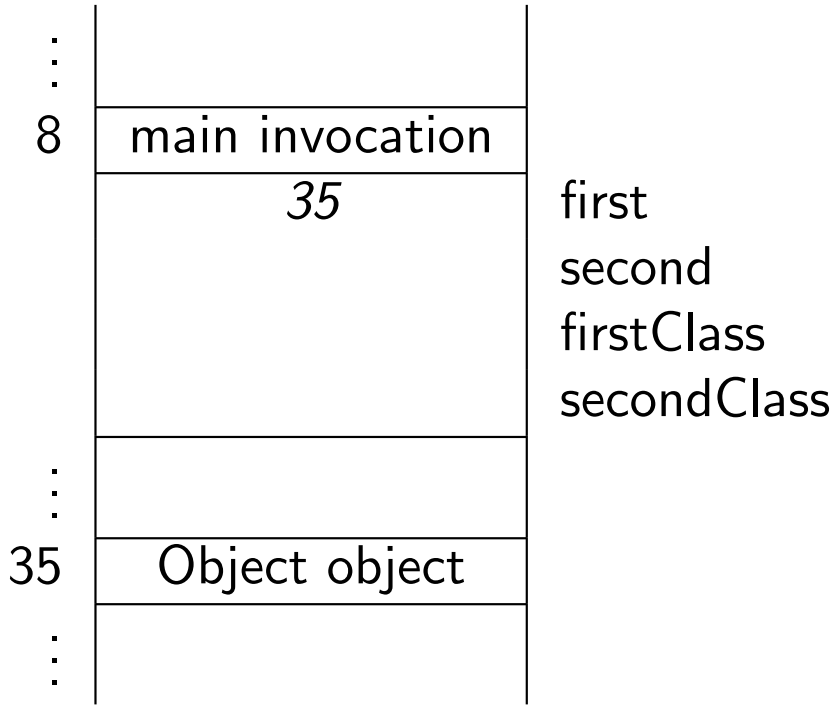
Memory diagrams

```
Object first = new Object();
Object second = new Object();
Class firstClass = first.getClass();
Class secondClass = second.getClass();
```



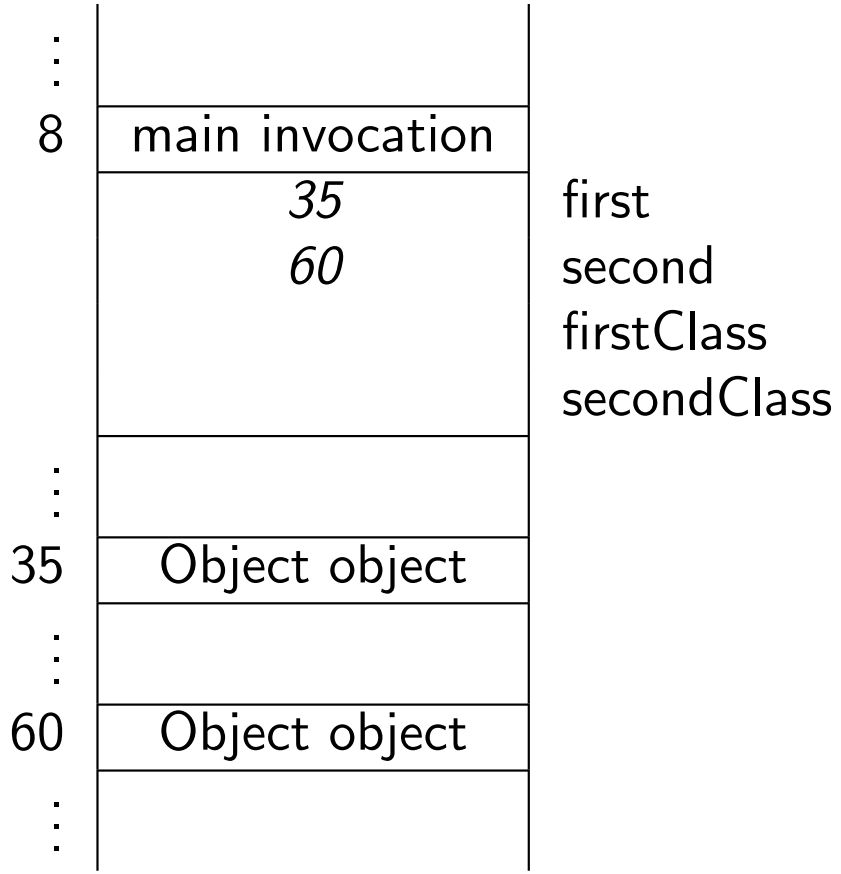
Memory diagrams

```
Object first = new Object();
Object second = new Object();
Class  firstClass  = first . getClass ();
Class  secondClass = second.getClass ();
```



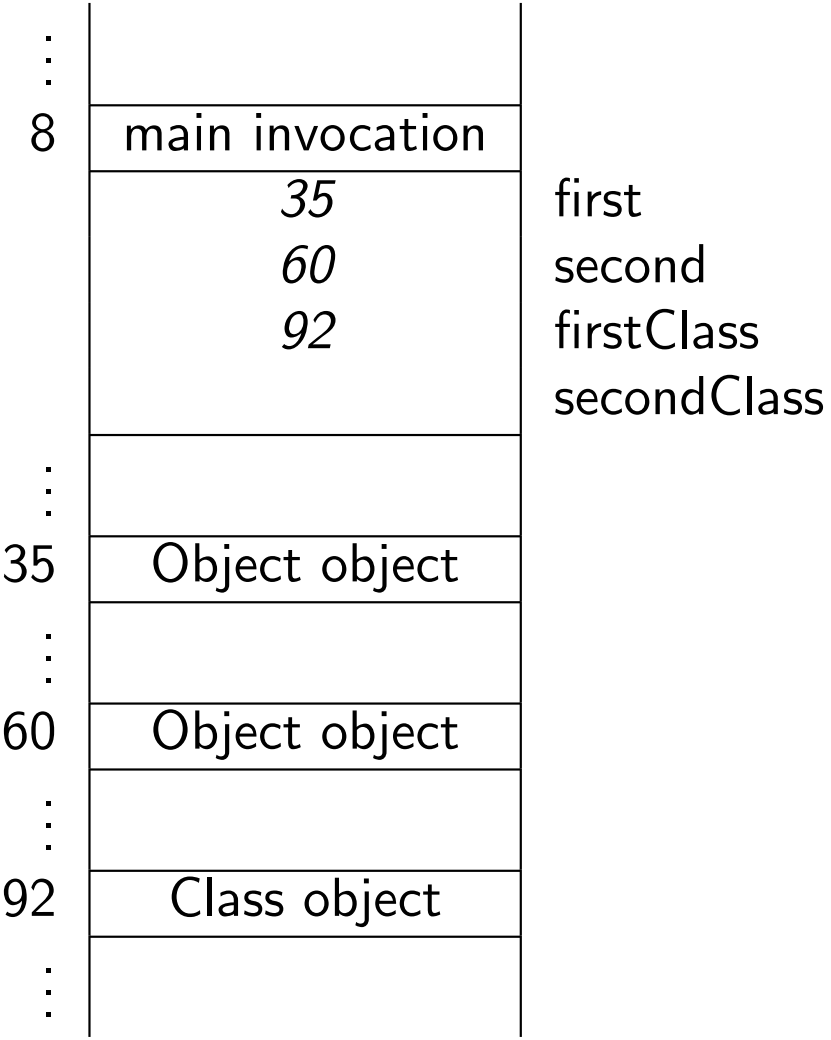
Memory diagrams

```
Object  first  = new Object();
Object second = new Object();
Class  firstClass  = first .getClass ();
Class secondClass = second.getClass ();
```



Memory diagrams

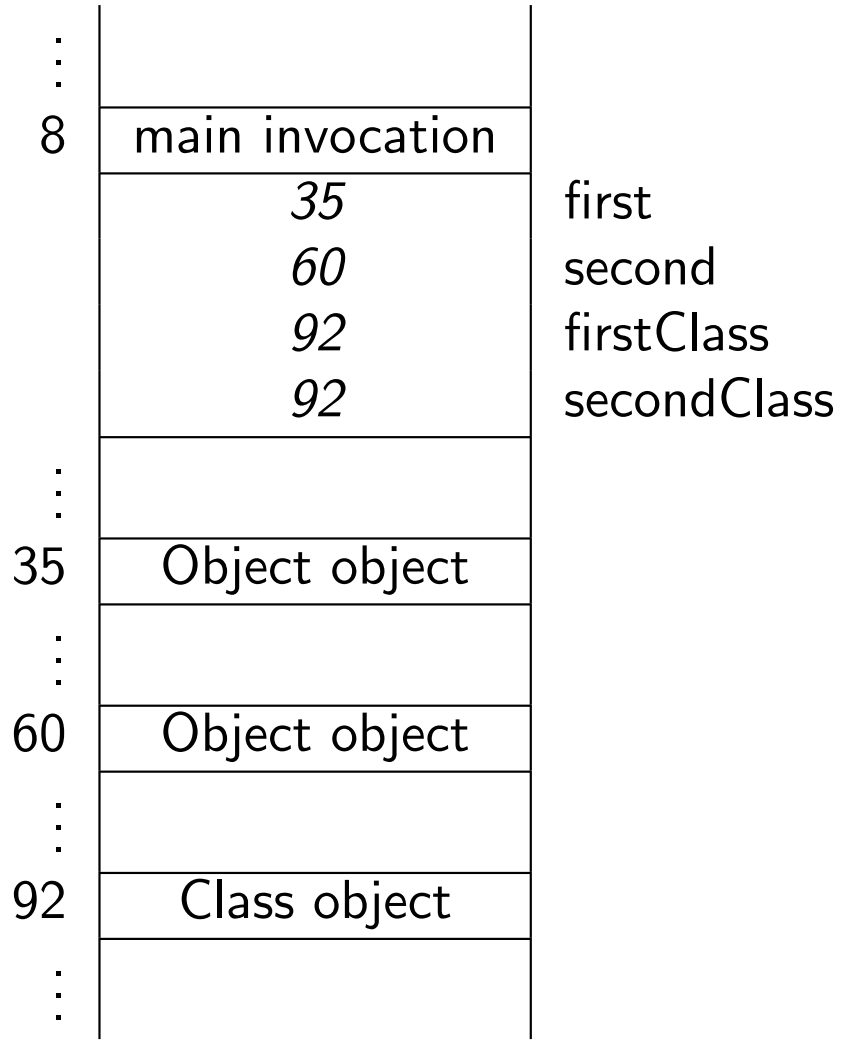
```
Object  first  = new Object();
Object second = new Object();
Class firstClass = first.getClass();
Class  secondClass  = second.getClass();
```



The `Class` object at address 92 in the unique `Class` object that represents the class `Object`.

Memory diagrams

```
Object   first   = new Object();
Object second = new Object();
Class   firstClass   =  first . getClass ();
Class secondClass = second.getClass();
```



equals of Rectangle

Question

What should `equals` return if `object` is not a `Rectangle`?

equals of Rectangle

Question

What should `equals` return if `object` is not a `Rectangle`?

Answer

`false`

equals of Rectangle

Question

What should `equals` return if `object` is not a `Rectangle`?

Answer

`false`

Question

How do we capture this in the body of the `equals` method?

equals of Rectangle

Question

What should `equals` return if `object` is not a `Rectangle`?

Answer

`false`

Question

How do we capture this in the body of the `equals` method?

Answer

```
if (this.getClass() != object.getClass())
{
   equal = false;
}
```

equals of Rectangle

Question

Assume that `object` is a `Rectangle` different from `null`. How do we check equality?

equals of Rectangle

Question

Assume that `object` is a `Rectangle` different from `null`. How do we check equality?

Answer

Check if the width and height are the same.

equals of Rectangle

```
boolean equal;
if (object != null && this.getClass() == object.getClass())
{
   Rectangle other = (Rectangle) object;
   equal = this.width == other.width
           && this.height == other.height;
}
else
{
   equal = false;
}
return equal;
```

equals of Rectangle

Question

Why does

```
if (object != null && this.getClass() == object.getClass())
```

never throw a `NullPointerException`?

equals of Rectangle

Question

Why does

```
if (object != null && this.getClass() == object.getClass())
```

never throw a `NullPointerException`?

Answer

If `object != null` then `this.getClass() == object.getClass()` is not evaluated.

equals

```
public boolean equals(Object object)
{
   boolean equal;
   if (object != null && this.getClass() == object.getClass
   {
      ... other = (...) object;
      equal = ...;
   }
   else
   {
      equal = false;
   }
   return equal;
}
```

Comparable interface

Question

What does

```
public Rectangle implements Comparable<Rectangle>
```

capture?

Comparable interface

Question

What does

```
public Rectangle implements Comparable<Rectangle>
```

capture?

Answer

The `Rectangle` class has to implement all methods specified in the `Comparable` interface.

Comparable interface

The interface `Comparable` imposes a total ordering on the objects of each class that implements it.

When we order `first` and `second`, either

- `first` is before/smaller than `second`,
- `first` and `second` are equal, or
- `first` is after/greater than `second`

Since there are three different results, we cannot use a `boolean` to represent it.

compareTo method

```
public int compareTo(T object)
```

- first is before/smaller than second
  `first.compareTo(second) < 0`
- first and second are equal
  `first.compareTo(second) = 0`
- first is after/greater than second
  `first.compareTo(second) > 0`

compareTo method for Integer

Question

```
public class Integer
{
   private int value;

   public int compareTo(Integer other)
   {
      return ...;
   }
}
```

Which expression using `this.value` and `other.value` is negative/zero/positive if `this.value` is smaller/equal/greater than `other.value`?

compareTo method for Integer

Question

```
public class Integer
{
   private int value;

   public int compareTo(Integer other)
   {
      return ...;
   }
}
```

Which expression using `this.value` and `other.value` is negative/zero/positive if `this.value` is smaller/equal/greater than `other.value`?

Answer

```
this.value - other.value
```

compareTo method for Rectangle

See API.

Question

To test the `getWidth` method, what does a test case consist of?

Testing

Question

To test the `getWidth` method, what does a test case consist of?

Answer

A `Rectangle` object.

Question

Which `Rectangle` object do we use?

Testing

Question

Which `Rectangle` object do we use?

Answer

Randomly chosen ones and boundary cases.

Question

What are the boundary cases?

Testing

Question

What are the boundary cases?

Answer

Width or height with value 0 or Integer.MAX_VALUE.