OBJECT ORIENTED PROGRAMMING
WEEK 3

6 FEB-11 FEB, 2022

Instructor:

Abdul Aziz
Assistant Professor
(School of Computing)
National University- FAST (KHI Campus)

ACKNOWLEDGMENT

- Publish material by Virtual University of Pakistan.

- Publish material by Deitel & Deitel.

- Publish material by Robert Lafore.

PRINCIPLES OF ENCAPSULATION

"Don't ask how I do it, but this is what I can do"

- The encapsulated object

"I don't care how, just do your job, and I'll do mine"

- One encapsulated object to another

ENCAPSULATING A CLASS

- Members of a class must always be declared with the minimum level of visibility.
- Provide *setters and getters* (also known as accessors/mutators) to allow *controlled* access to private data.

CONSTRUCTOR

- A member function like any other function. But, special in nature.
- It has same name as class name.
- Called automatically when an object is created.
- Only called once in a lifeline of an object.
- Used to assigned memory to the object.
- Never return a value.
- May have parameters.
- Every class should have at least one constructor.
  - If you don't write constructor, compiler will generate the **default constructor**
- Constructors are usually declared public.
  - Constructor can be declared as private → You can't use it outside the class.

Declaring Constructors

- **Syntax:**

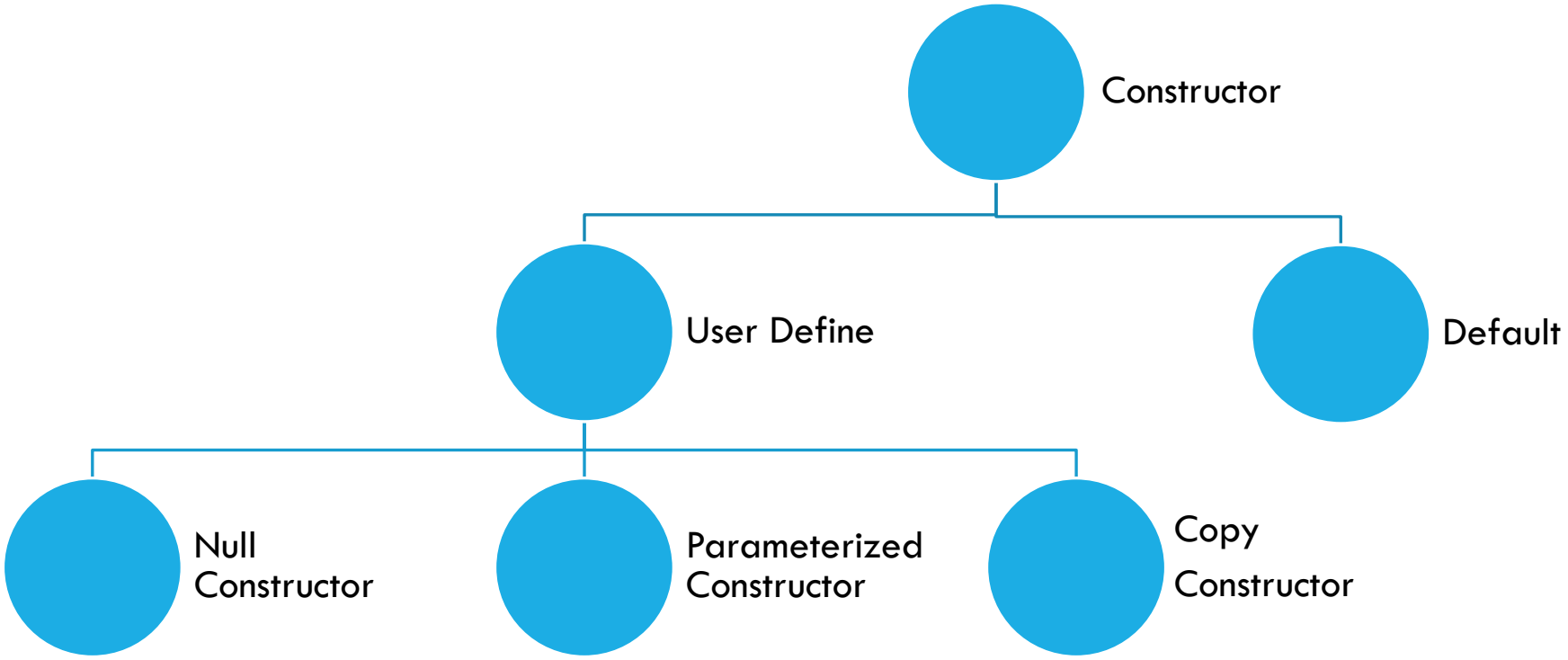
```
[<modifier>]<class_name>( <argument>*){
    <statement>*
}
```

```
public class Date {
  private int year, month, day;

  public Date( int y, int m, int d)     {
     if( verify(y, m, d) ){
       year = y; month = m; day = d;
     }
  }

  private boolean verify(int y, int m, int d){
    //...
  }
}
```

TYPES OF CONSTRUCTORS



The Default Constructors

- There is always **at least one constructor** in every class.
- If the programmer does not supply any constructors, the default constructor is generated by the compiler
  - The default constructor takes no argument
  - The default constructor's body is empty

```
public class Date {
  private int year, month, day;

  public Date( ){
  }
}
```

PARAMETERIZED CONSTRUCTOR

A constructor that initialized an object with user define values.

```
public class Date {
        private int year, month, day;
        public Date( ){}

        public Date (int y, int m, int d ){
                year=y;
                month=m;
                day=d;
        }
}
```

COPY CONSTRUCTOR

A constructor that creates a copy of any given object (Clone)

The constructor receive an object as reference and create a new object that is similar to the given object.

```
public class Date {
        private int year, month, day;
        public Date( ){}

        public Date ( Date d){
                year=d. year;
                month=d. month;
                day=d.day;
        }
}
```

STYLE 1

```
public class Date {
        private int year, month, day;
        public Date( ){}

        public Date ( Date d){
                year=d. year;
                month=d. month;
                day=d.day;
        }
 }
```

STYLE 2

```
public class Date {
        private int year, month, day;
        public Date( ){}

        public Date (int y, int m, int d ){
                year=y;
                month=m;
                day=d;
        }

        public Date ( Date d){
                this(d. year, d. month, d.day);
        }
 }
```

STYLE 3

```
public class Date {
        private int year, month, day;
        public Date( ){}

        public Date ( Date d){
                this.year=d. year;
                this.month=d. month;
                this.day=d.day;
        }
}
```

CALLING CONSTRUCTORS OF OBJECTS

- Objects are **instances** of classes
- Are **allocated on the heap** by using the new operator
- Constructor is invoked automatically on the new object

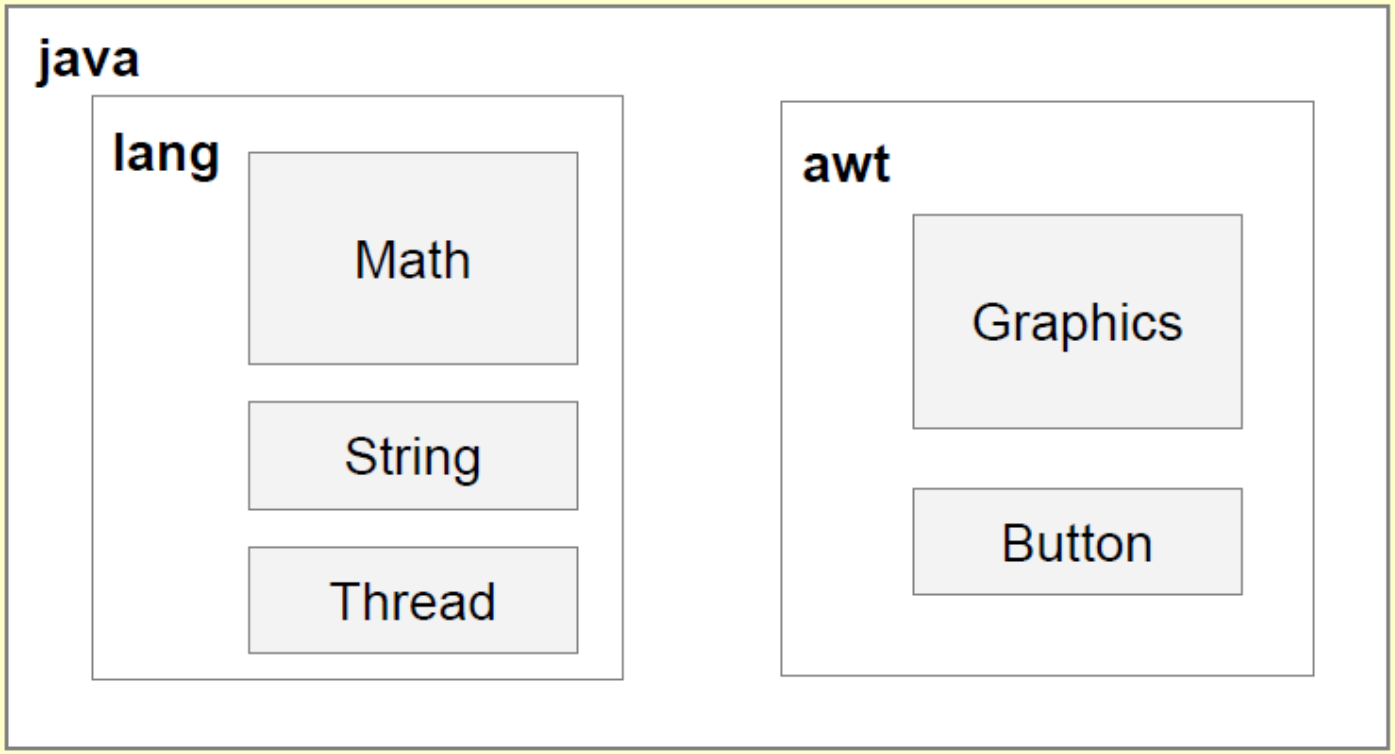
```
Date d = new Date();

Date d1 = new Date( 2016, 9, 23);

Date d2 = new Date(d1);
```

Packages

- Help manage large software systems
- Contain
  - Classes
  - Sub-packages



The package statement

- Syntax:

```
package <top_pkg_name>[.<sub_pkg_name>]*;
```

- Examples:

```
package java.lang;

public class  String{
    //...
}
```

- statement **at the beginning** of the source file
- only **one package declaration** per source file
- if **no package name** is declared → the class is placed into the **default package**

The `import` statement

- Syntax:

```
package <top_pkg_name>[.<sub_pkg_name>]*;
```

- Usage:

```
import <pkg_name>[.<sub_pkg_name>]*.*;
```

- Examples:

```
import java.util.List;
import java.io.*;
```

-precedes all class declarations
-tells the compiler **where to find classes**

Java Types

- **Primitive (8)**
  - Logical: `boolean`
  - Textual: `char`
  - Integral: `byte, short, int, long`
  - Floating: `double, float`
- **Reference**
  - All others

Logical - `boolean`

- Characteristics:
  - Literals:
    - `true`
    - `false`
  - Examples:
    - `boolean cont = true;`
    - `boolean exists = false;`

Textual - `char`

- Characteristics:
  - Represents a 16-bit Unicode character
  - Literals are enclosed in single quotes (' ')
  - Examples:
    - `'a'` - the letter `a`
    - `'\t'` - the `TAB` character
    - `'\u0041'` - a specific Unicode character (`'A'`) represented by 4 hexadecimal digits

Integral – `byte, short, int, and long`

- Characteristics:
  - Use three forms:
    - Decimal: `67`
    - Octal: `0103` (`1x8^2+0x8^1+3x8^0`)
    - Hexadecimal: `0x43`
  - Default type of literal is `int`.
  - Literals with the `L` or `l` suffix are of type `long`.

Floating Point – `float` and `double`

- Characteristics:
  - **Size:**
    - `float` – 4 byte
    - `double` – 8 byte
  - Decimal point
    - `9.65` (`double`, **default type**)
    - `9.65`**`f`** or `9.65`**`F`** (`float`)
    - `9.65`**`D`** or `9.65`**`d`** (`double`)
  - Exponential notation
    - `3.41E20` (`double`)

Java ReferenceTypes

```
public class  MyDate{
    private int day = 26;
    private int month = 9;
    private int year = 2016;

    public MyDate( int day, int month, int year){
    ...
    }
}
```

```
MyDate date1 = new MyDate(20, 6, 2000);
```

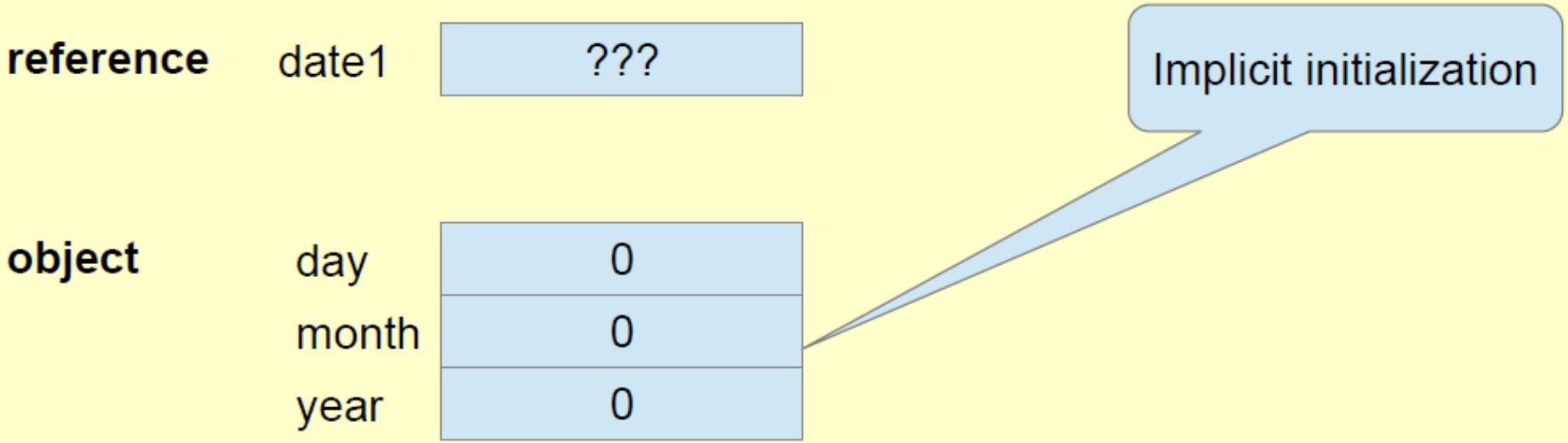
1) Memory is allocated for the object
2) Explicit attribute initialization is performed
3) A constructor is executed
4) The **object reference** is returned by the `new` operator

```
date1 =  object reference
```

5) The reference is assigned to a variable

Memory is allocated for the object

```
MyDate date1 = new MyDate();
```



Executing the constructor

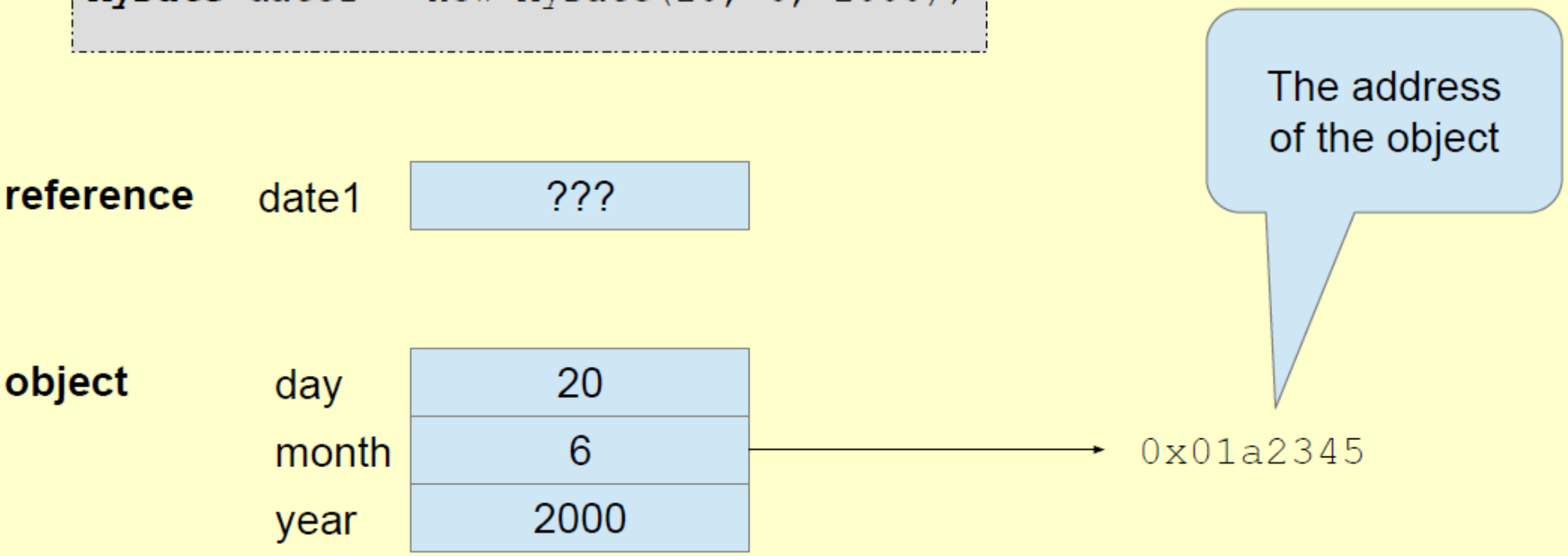
```
MyDate date1 = new MyDate(20, 6, 2000);
```

reference	date1	???

object	day	20
	month	6
	year	2000

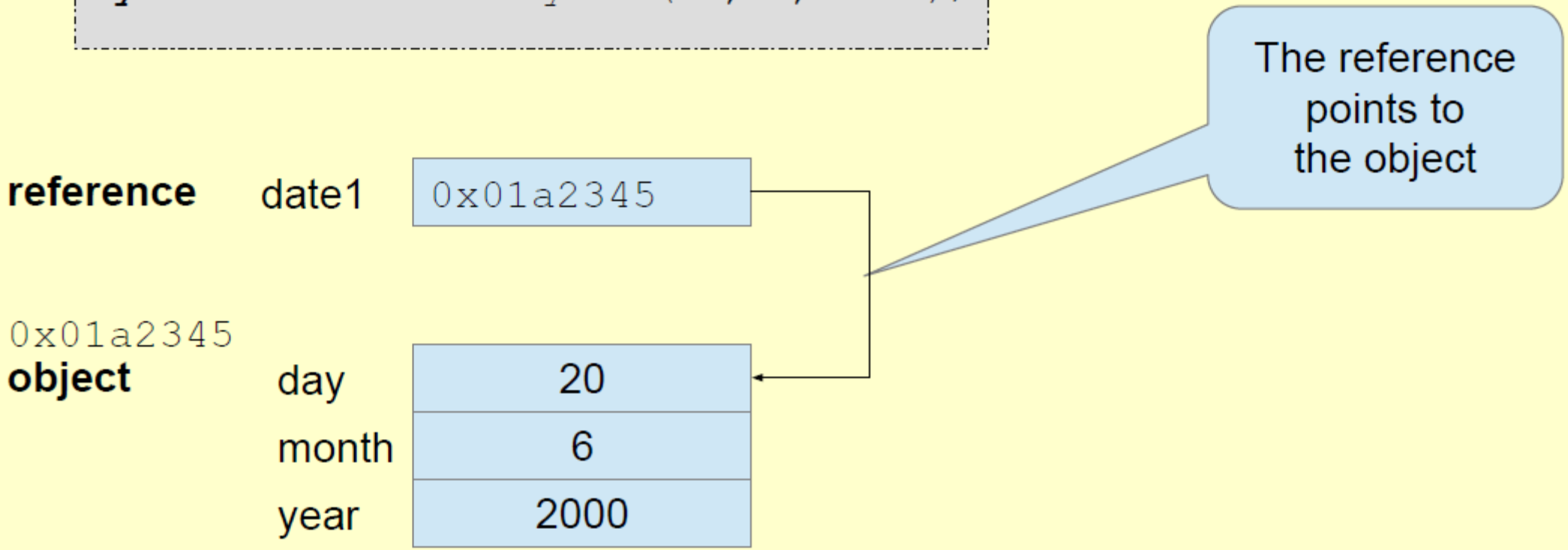
The object reference is returned

```
MyDate date1 = new MyDate(20, 6, 2000);
```



The reference is assigned to a variable

```
MyDate date1 = new MyDate(20, 6, 2000);
```



Assigning References

- Two variables refer to a single object

```
MyDate date1 = new MyDate(20, 6, 2000);
MyDate date2 = date1;
```