How to Make an Object Inspector in Java

M Akif Eyler

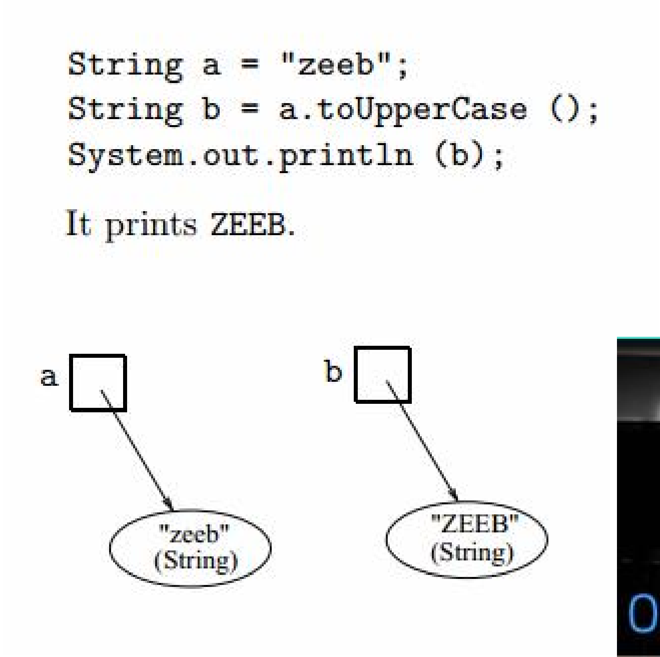
Marmara University

June 2014

Outline

- Structure of an object (in Java)
- insp.jar – a simple object inspector
  “How it tastes”
- `class sun.misc.Unsafe` – the kitchen
  “How it’s made”

Two objects, two references



```
String a = "zeeb";
String b = a.toUpperCase ();
System.out.println (b);
```

It prints ZEEB.

Ref: MIT OCW Course 6.170

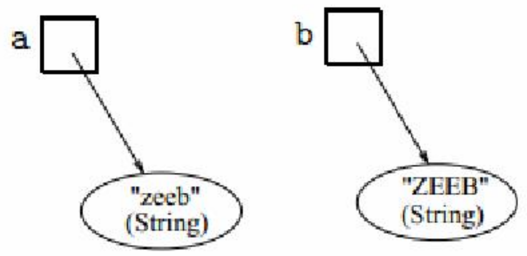
ALWAYS OPEN
OCW is always open. Here when you need it most.
MITOPENCOURSEWARE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Two objects – again

two references:

```
String a = "zeeb";
String b = a.toUpperCase ();
System.out.println (b);
```

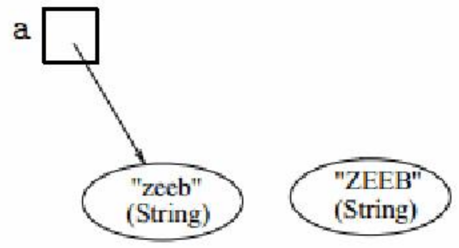
It prints ZEEB.



one reference:

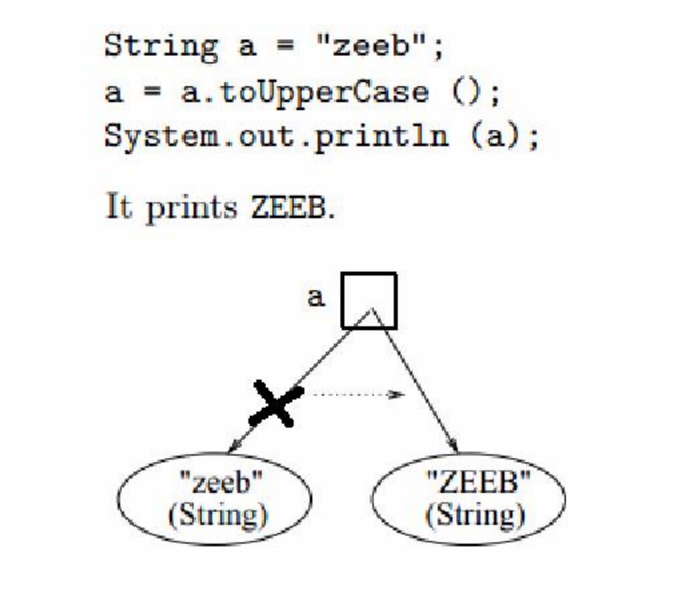
```
String a = "zeeb";
a.toUpperCase ();
System.out.println (a);
```

It prints zeeb.



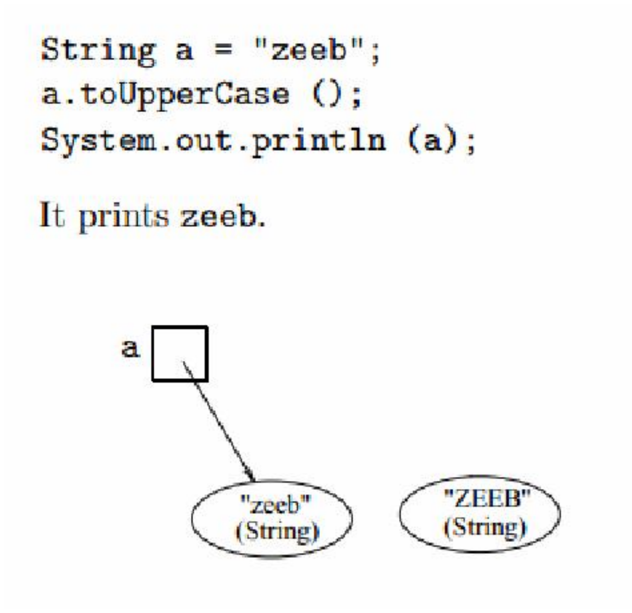
String objects are immutable,
they cannot be modified

Two objects, one reference



```
String a = "zeeb";
a = a.toUpperCase ();
System.out.println (a);
```

It prints ZEEB.



```
String a = "zeeb";
a.toUpperCase ();
System.out.println (a);
```

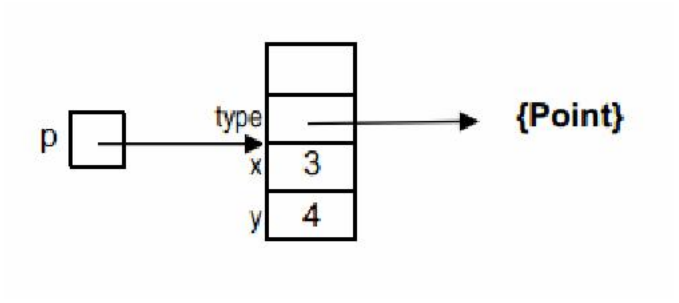
It prints zeeb.

String objects are immutable, they cannot be modified

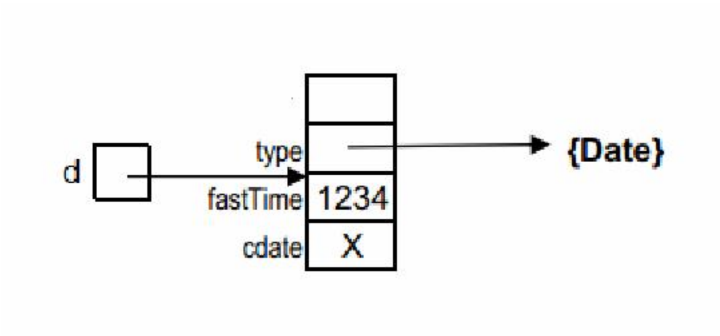
Structure of an object

```
import java.awt.Point;
import java.util.Date;
```

```
p = new Point(3, 4);
```

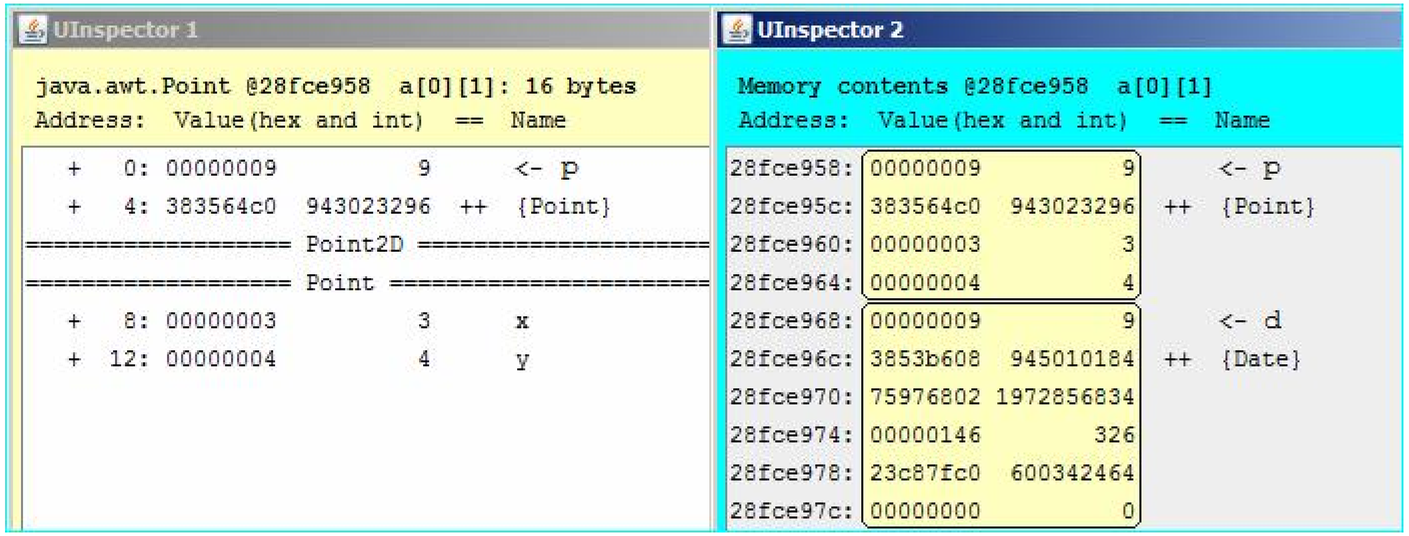


```
d = new Date(); //represents “now”
```



Structure of an object

```
p = new Point(3, 4);
d = new Date();
```



UInspector 1

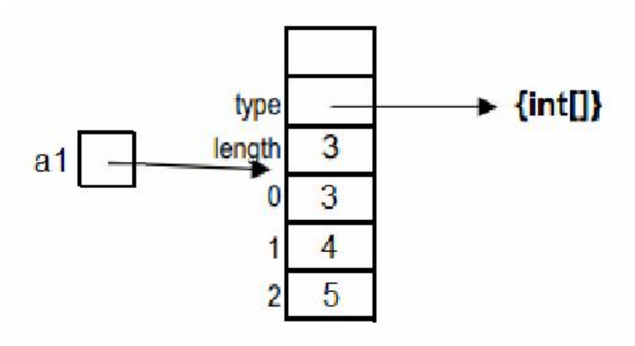
```
java.awt.Point @28fce958  a[0][1]: 16 bytes
Address:  Value(hex and int)  ==  Name
  +   0: 00000009          9     <- p
  +   4: 383564c0  943023296  ++  {Point}
=================== Point2D =====================
=================== Point =======================
  +   8: 00000003          3      x
  +  12: 00000004          4      y
```

UInspector 2

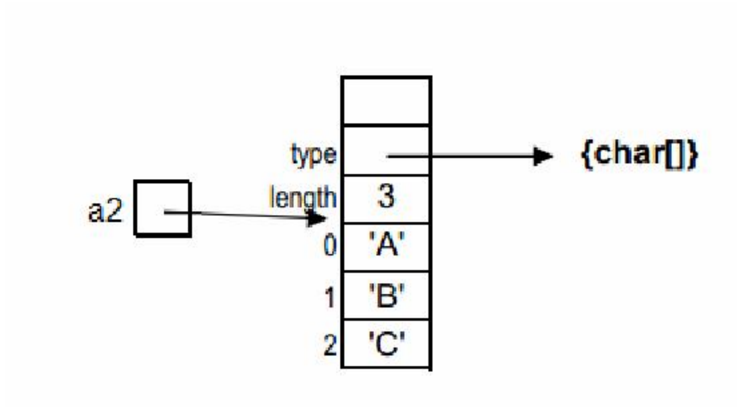
```
Memory contents @28fce958  a[0][1]
Address:  Value(hex and int)  ==  Name
28fce958: 00000009          9      <- p
28fce95c: 383564c0  943023296  ++  {Point}
28fce960: 00000003          3
28fce964: 00000004          4
28fce968: 00000009          9      <- d
28fce96c: 3853b608  945010184  ++  {Date}
28fce970: 75976802 1972856834
28fce974: 00000146        326
28fce978: 23c87fc0  600342464
28fce97c: 00000000          0
```

Structure of an array

```
int[] a1 = { 3, 4, 5 };
```



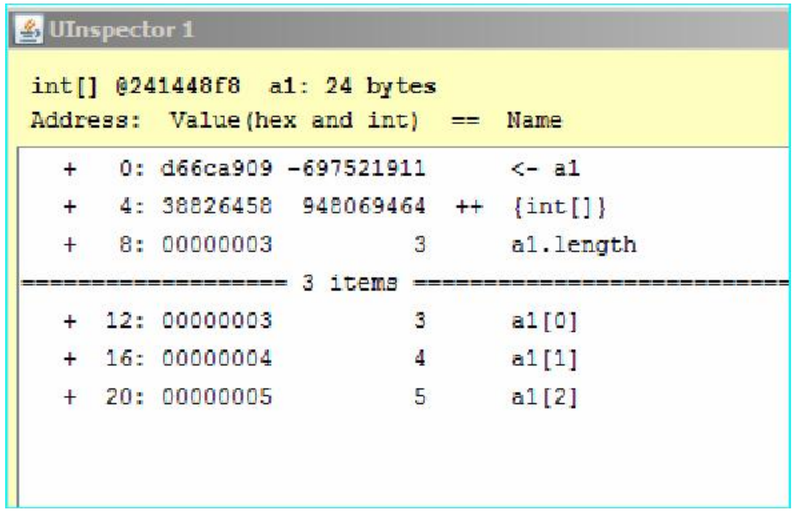
```
char[] a2 = { 'A', 'B', 'C' };
```



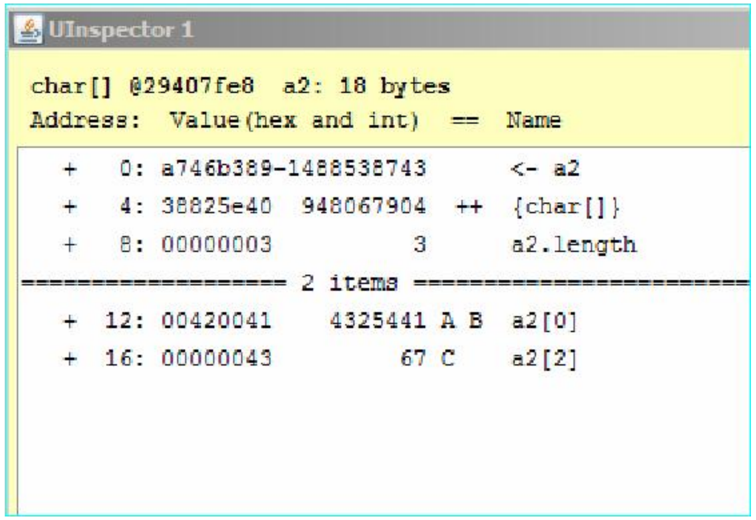
Structure of an array

```
int[] a1 = { 3, 4, 5 };
```

```
char[] a2 = { 'A', 'B', 'C' };
```


UInspector 1

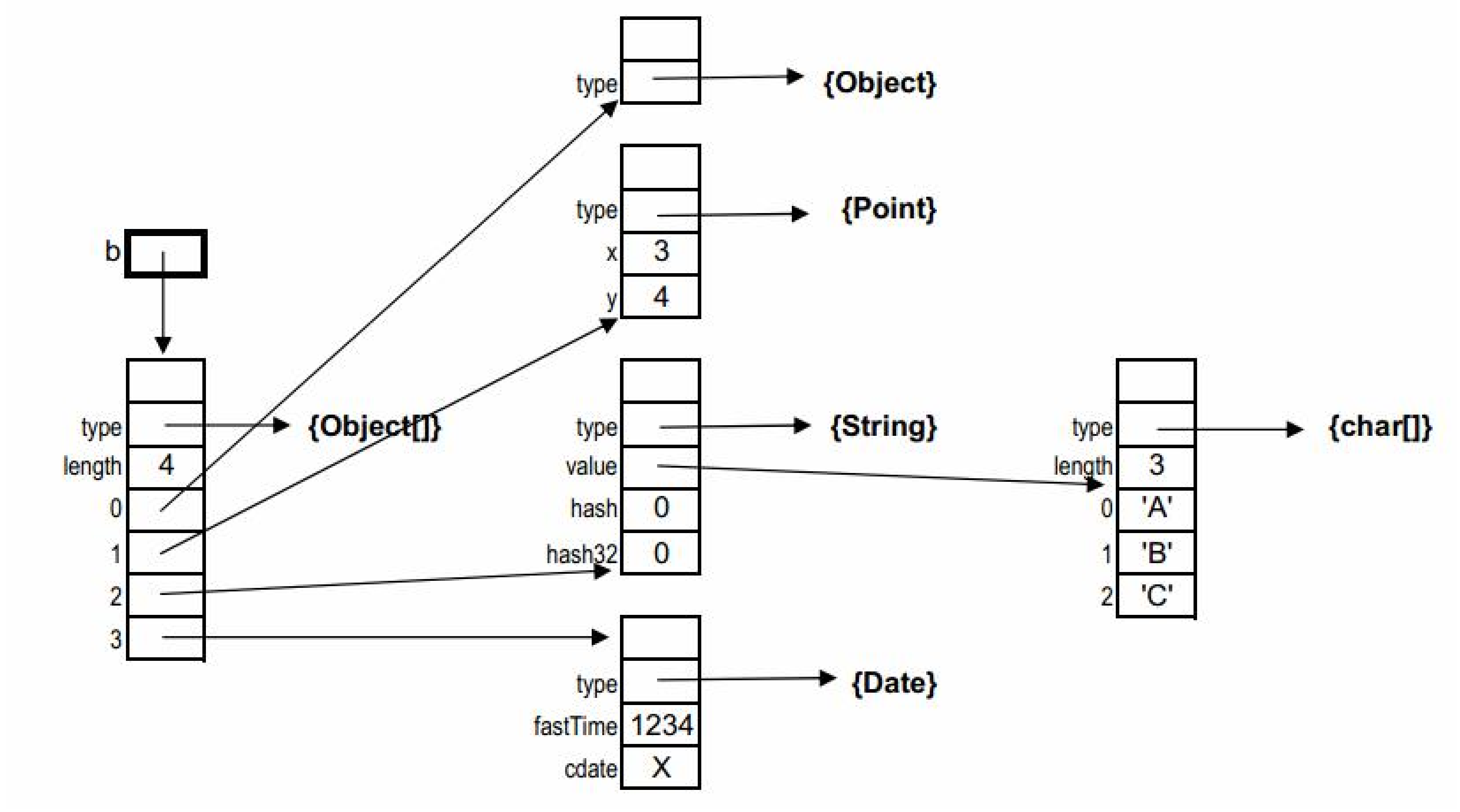
```
int[] @241448f8  a1: 24 bytes
Address:  Value(hex and int)  ==  Name
  +   0: d66ca909 -697521911      <- a1
  +   4: 38826458  948069464  ++  {int[]}
  +   8: 00000003          3      a1.length
=================== 3 items ============================
  +  12: 00000003          3      a1[0]
  +  16: 00000004          4      a1[1]
  +  20: 00000005          5      a1[2]
```


UInspector 1

```
char[] @29407fe8  a2: 18 bytes
Address:  Value(hex and int)  ==  Name
  +   0: a746b389-1488538743      <- a2
  +   4: 38825e40  948067904  ++  {char[]}
  +   8: 00000003          3      a2.length
=================== 2 items ========================
  +  12: 00420041    4325441 A B  a2[0]
  +  16: 00000043         67 C    a2[2]
```

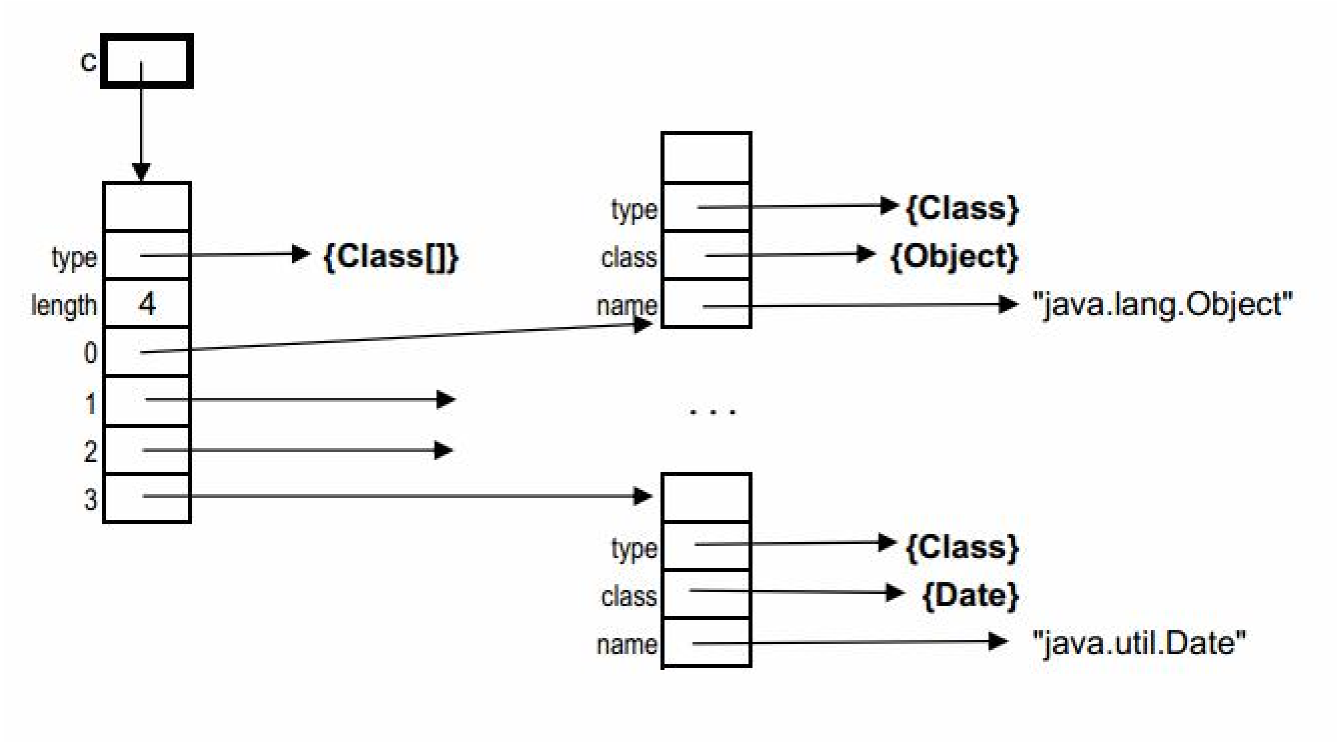
Structure of an array

```
Object[] b = { new Object(), new Point(3, 4), "ABC", new Date() };
```



Structure of an array

```
Class[]  c = { Object.class, Point.class, String.class, Date.class };
```



Now… Enter the kitchen…

sun.misc.Unsafe -- GrepCode Class Source

Java source code that includes Javadoc comments

Now… Enter the kitchen…

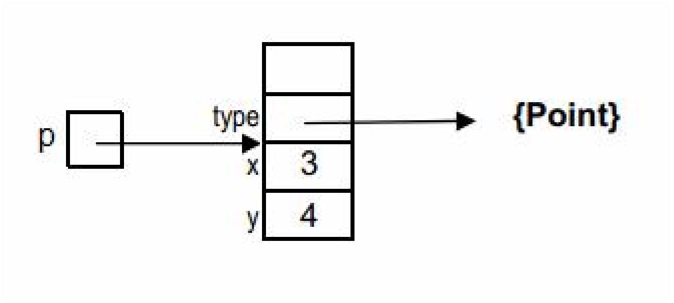
sun.misc.Unsafe -- GrepCode Class Source

With `sun.misc.Unsafe`, there is an alternative to low-level programming on the Java plarform using a Java API, even though this alternative is discouraged†. [...]
Therefore, it is time to have a look, especially since the functionality of `sun.misc.Unsafe` is considered to become part of Java's public API in Java 9.
http://java.dzone.com/articles/understanding-sunmiscunsafe

† *In general, writing java programs that rely on* `sun.*` *is risky: those classes are not portable, and are not supported.*
http://www.oracle.com/technetwork/java/faq-sun-packages-142232.html

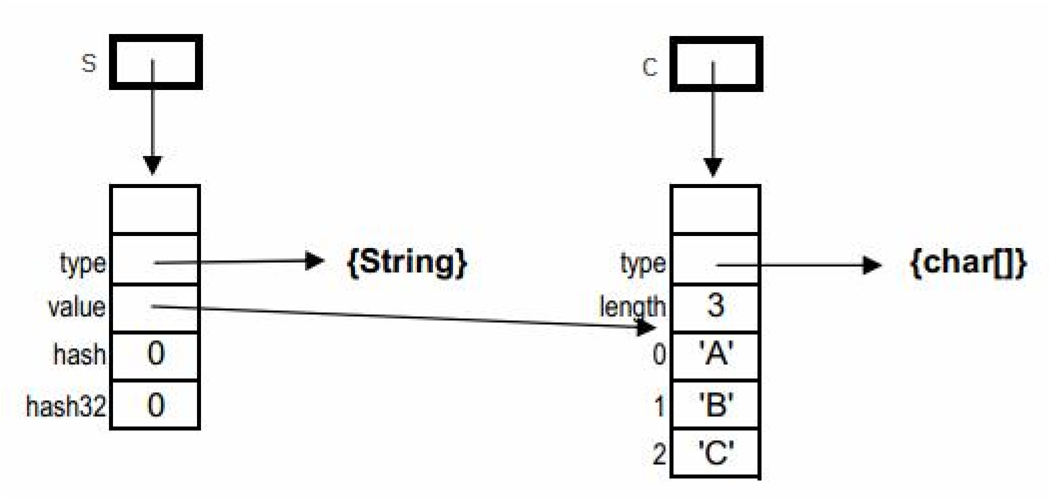
Using the Unsafe

```
//get the singleton instance using reflection
U = Unsafe.theUnsafe;
//32-bit JVM – 4 bytes per word
U.addressSize(); //--> 4
//start with any object
Point p = new Point(3, 4);
//get and set field values
U.getInt(p, 8); //--> 3
U.getInt(p,12); //--> 4
U.putInt(p,12, 99); // 4 becomes 99
```



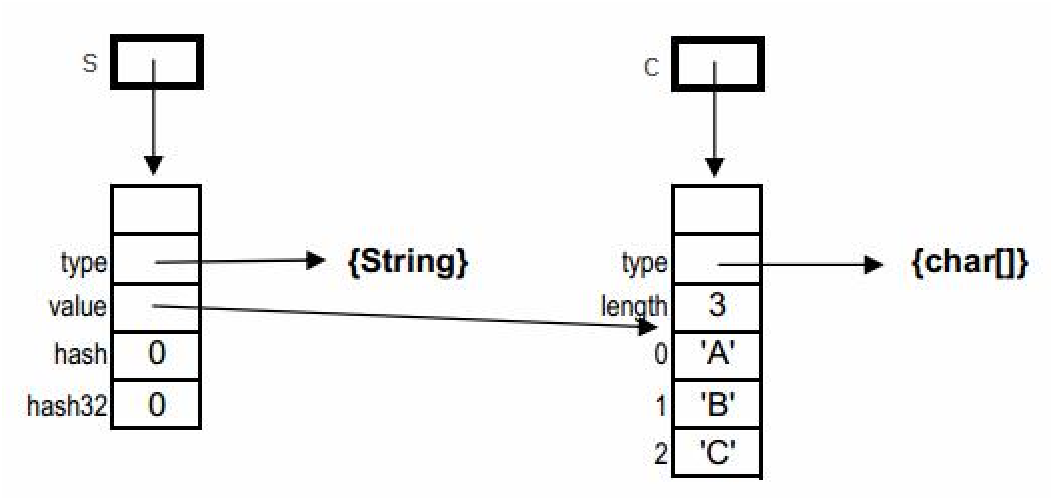
Using the Unsafe

```
//a String object
String s = "ABC";
//get field values
char[] c = U.getObject(s, 8); //--> value
int pc = U.getInt(s, 8); //--> @value
U.getInt(s,12); //--> hash
U.getInt(c, 8); //--> length
U.getChar(c, 12); //--> c[0]
```



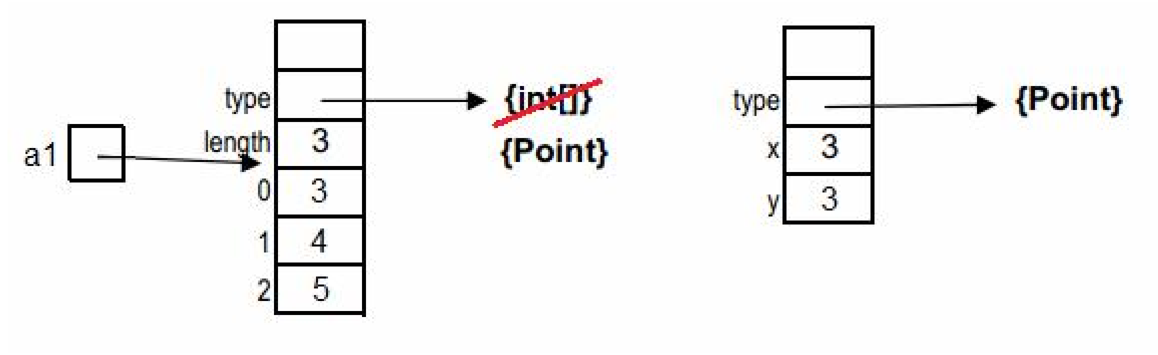
Modify “immutable” objects

```
//a String object s and its value c
String s = "ABC";
char[] c = U.getObject(s, 8); //--> value
//set field values
U.putChar(c, 12, 'm'); //--> c[0] becomes 'm'
U.putInt(c, 8, 1); //--> length becomes 1
//s contains "m"
```



Magic: Modify object type!

```
//caution - dangerous waters
//our old friends p and a1
Point p = new Point(3, 4);
int t = U.getInt(p, 4); //--> type
int[] a1 = { 3, 4, 5 };
U.putInt(a1, 4, t); // type is modified
//a1 is now a Point object – equal to new Point(3, 3)
```


a1
type
length
0
1
2
3
3
4
5
~~{int[]}~~
{Point}
type
x
y
3
3
{Point}