CPSC-224
Software Development
JVM and Memory Allocation

Yu Wang

wangy2@gonzaga.edu

Jan 29, 2025

Announcement

- Quiz time: Jan 31 (Bring your computer)
- Homework1 (Check the updates on Canvas)
  - CPSC 224 - GitHub Classroom Assignment Instructions
  - Deadline for submission: Feb. 12

Daily Attendance (01)

- Scan the QR Code for yourself

Daily Attendance (02)

- Scan the QR Code for yourself

Review - Last Class

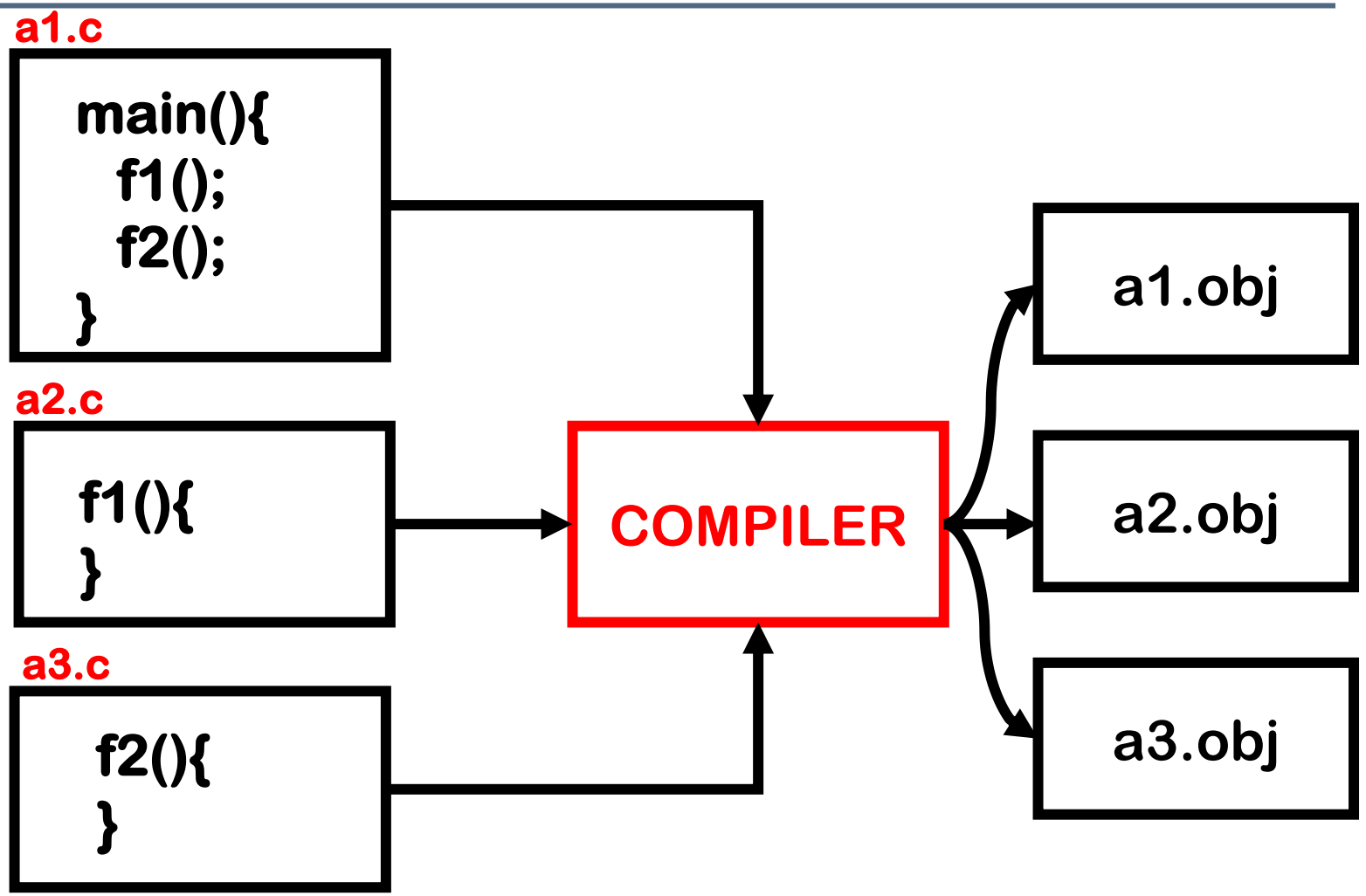
GONZAGA UNIVERSITY

- ✓ We learned – Escape Sequences, Arrays, Multidimensional Arrays and how to access elements, Constant and Casting.
- ✓ We learned – Class and Object, how they work together, and how to create class and object

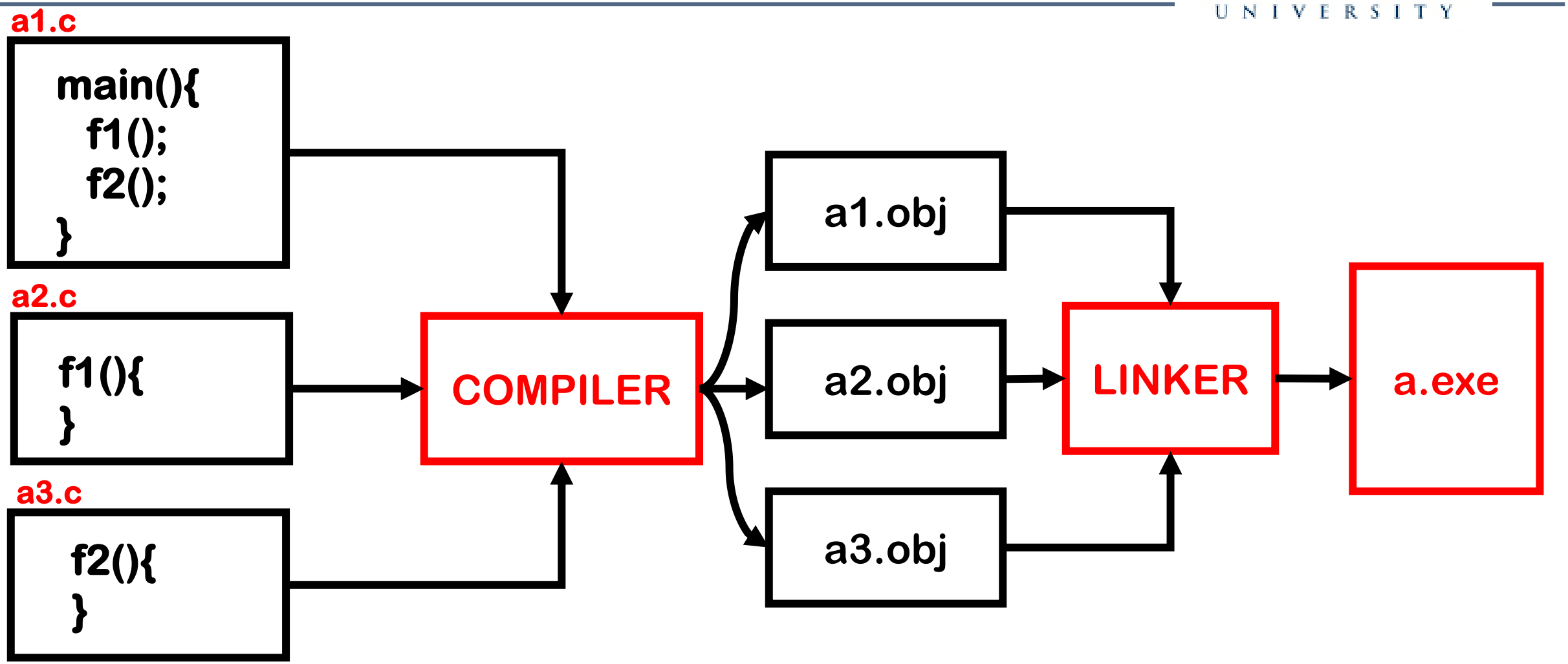
Java Virtual Machine (JVM)

- A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode.
- JVM(Java Virtual Machine): Executes bytecode.
- JRE (Java Runtime Environment): Provides the JVM and libraries needed to run Java applications.
- JDK (Java Development Kit): Includes the JRE, JVM, and tools like javac (compiler) to develop and run Java programs.

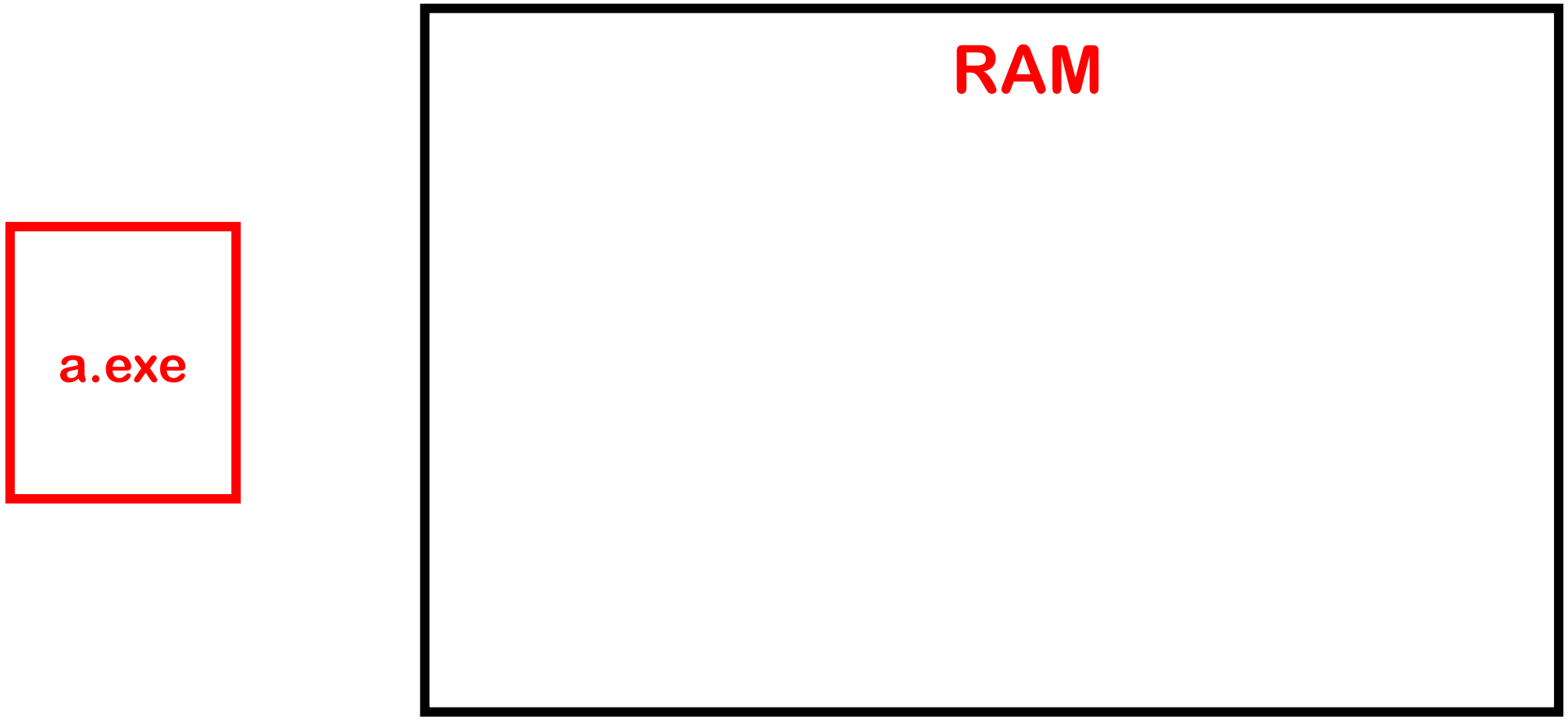
Java Virtual Machine (JVM) – C Process



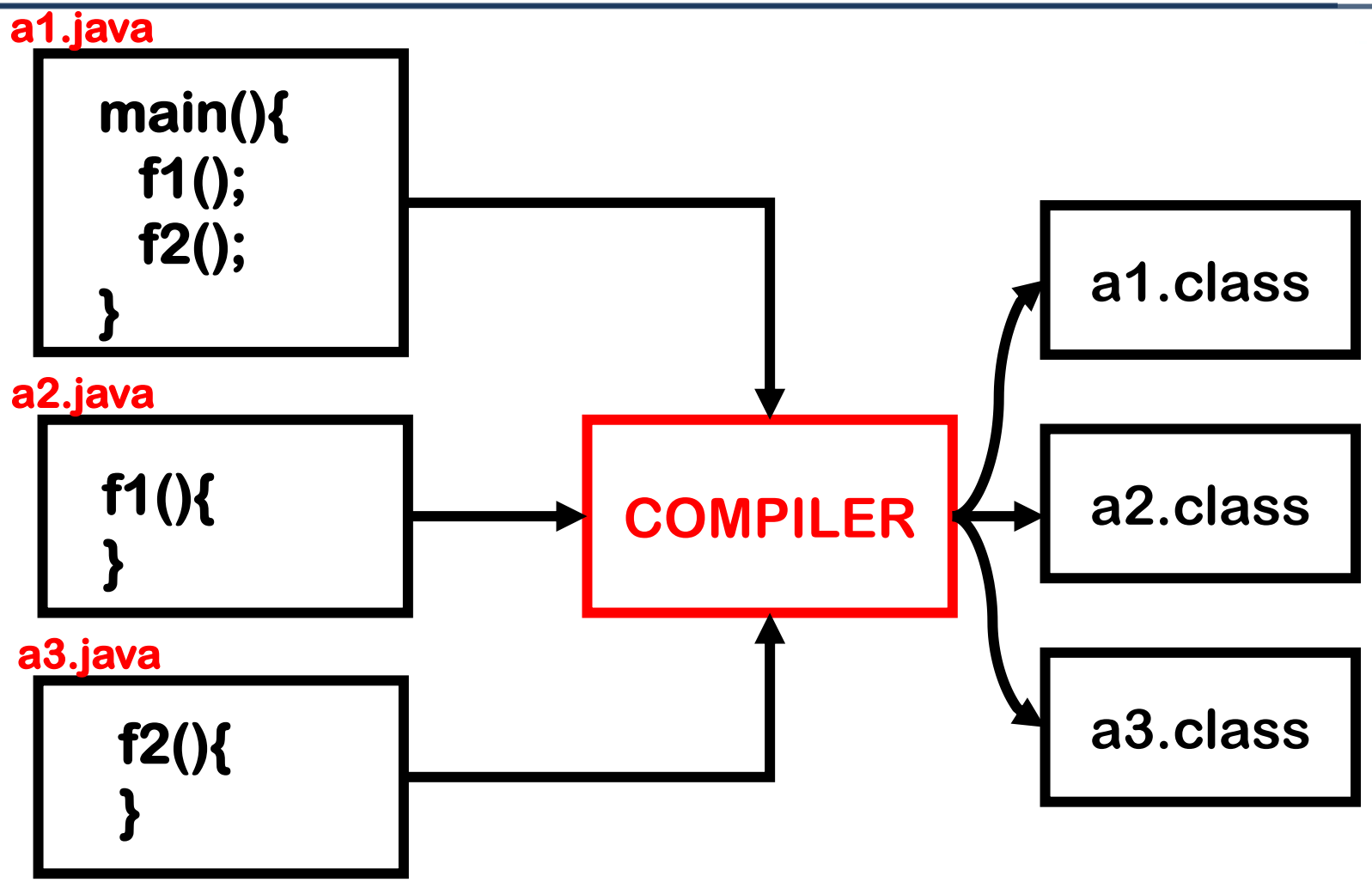
Java Virtual Machine (JVM) – C Example



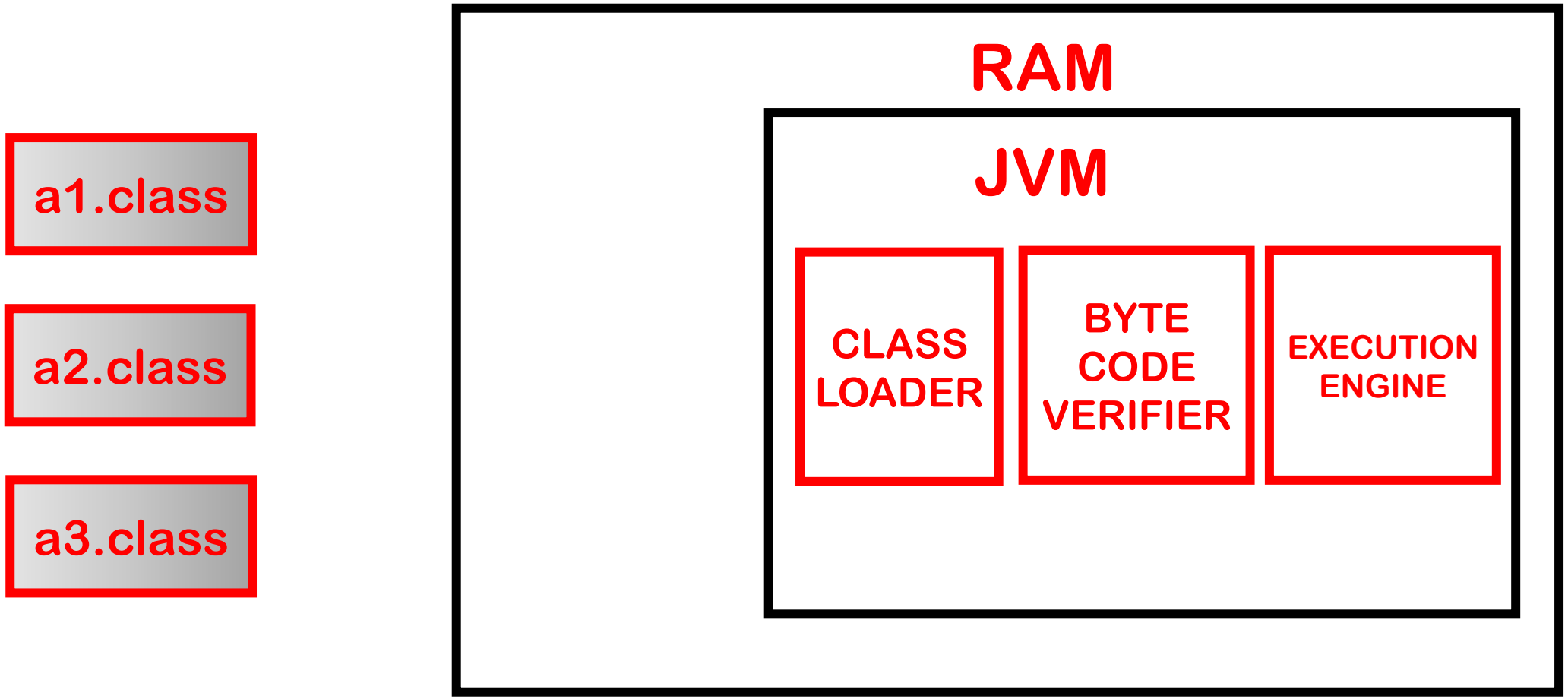
Java Virtual Machine (JVM) – C Process



Java Virtual Machine (JVM) – Java Process

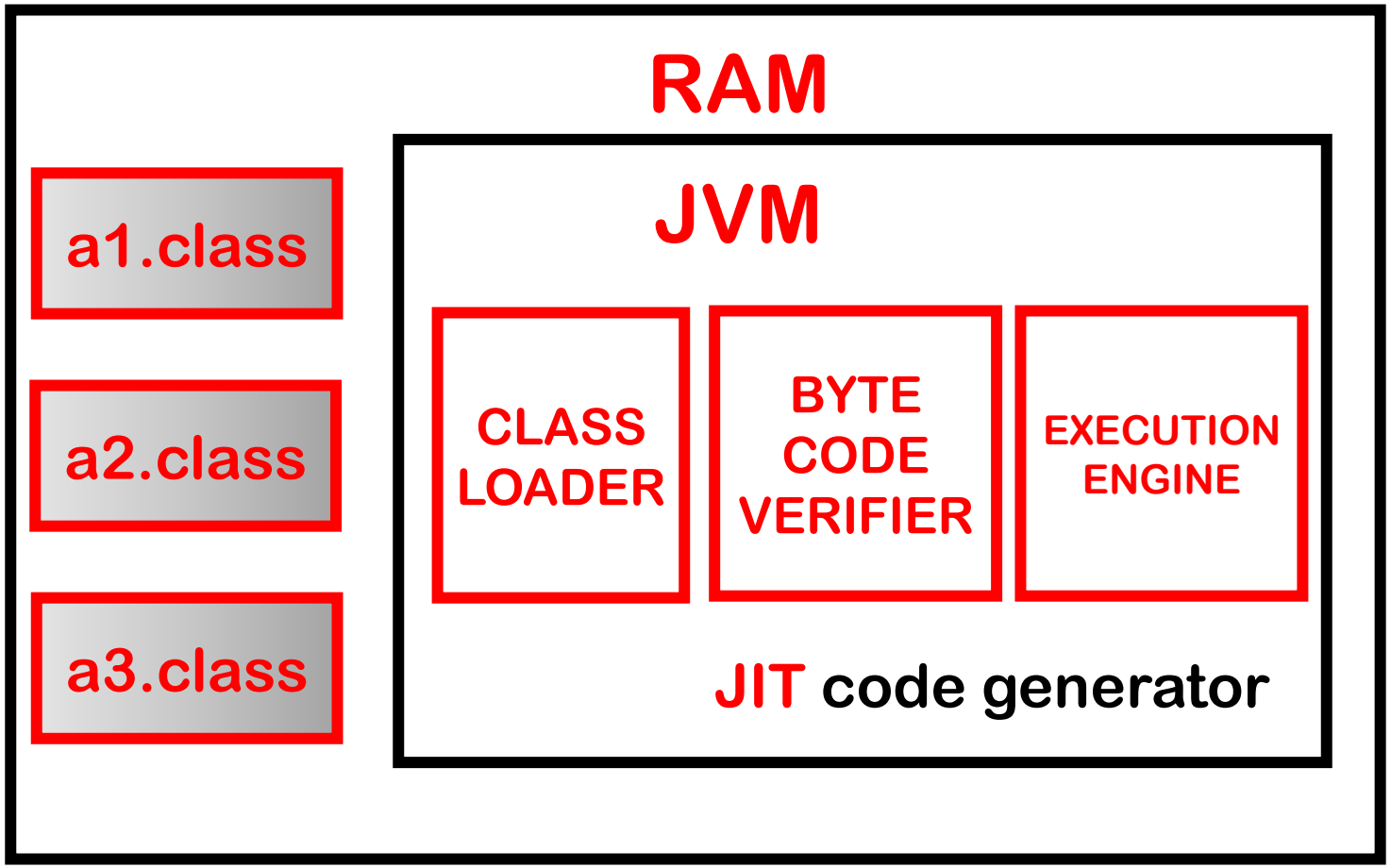


Java Virtual Machine (JVM) – Java Process

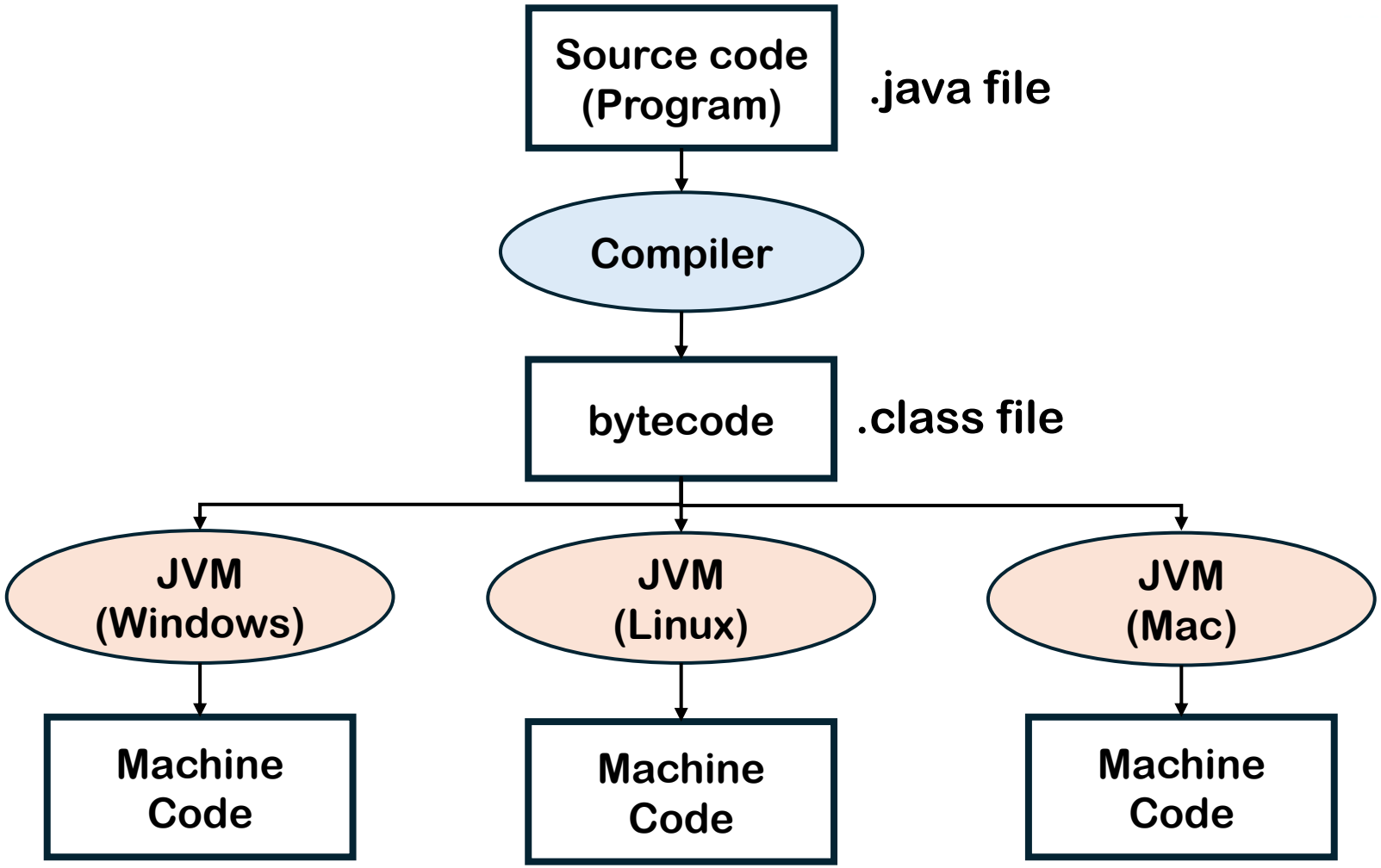


Java Virtual Machine (JVM) – Java Process

- JIT -The JIT (Just-In-Time) code generator is a critical component of the Java Virtual Machine (JVM) that improves the performance of Java applications by converting bytecode into native machine code at runtime .



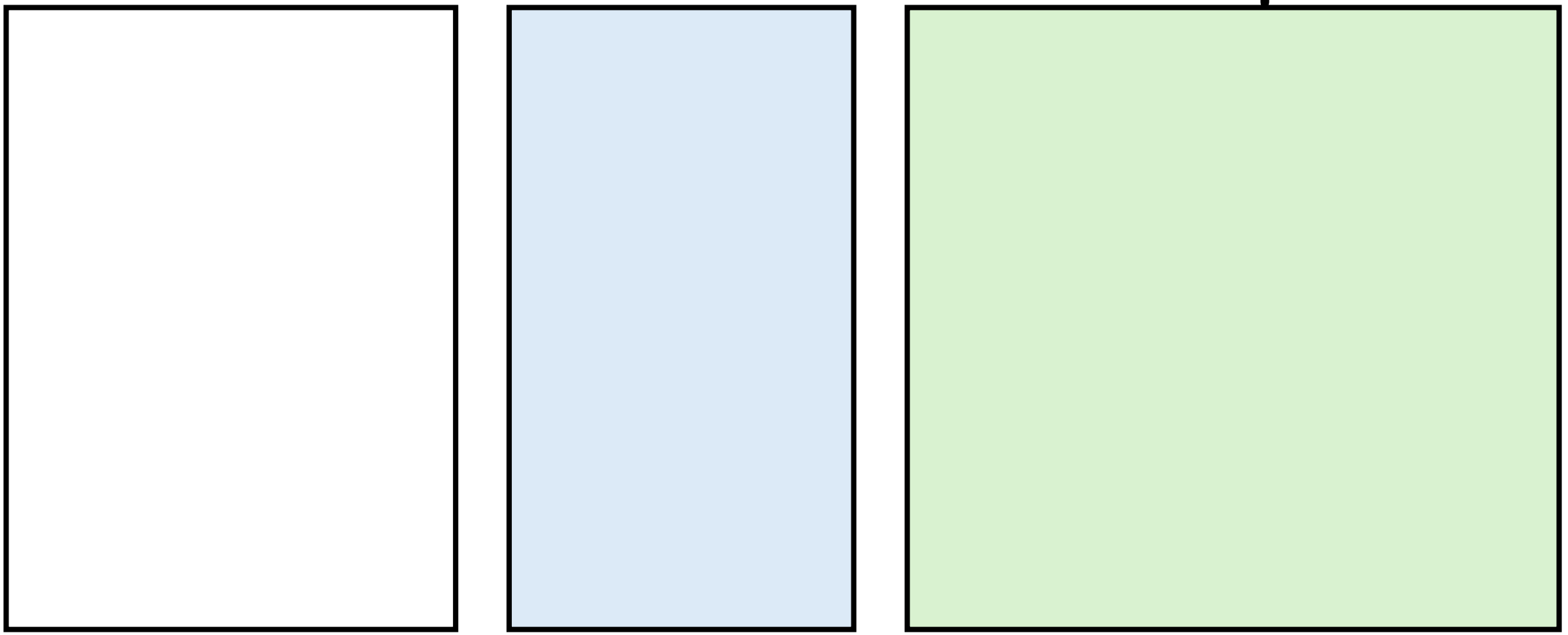
What is Java Bytecode?



Memory Allocation

- All data for primitive type variables is stored on the stack
- For reference types, the stack holds a pointer to the object on the heap
- When setting a reference type variable equal to another reference type variable, a copy of only the pointer is made
- Certain object types can not be operated on the heap

Memory Allocation



Memory Allocation

GONZAGA UNIVERSITY

Code

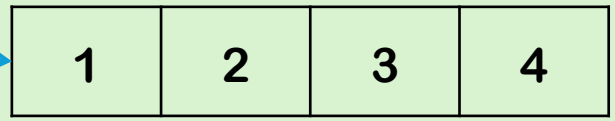
```
int a = 3;
int b = a;
b = 100;
Int [] c = {1,2,3,4};
Int [] d = c;
```

Stack

d =
c =

b = 100;
a = 3;

Heap



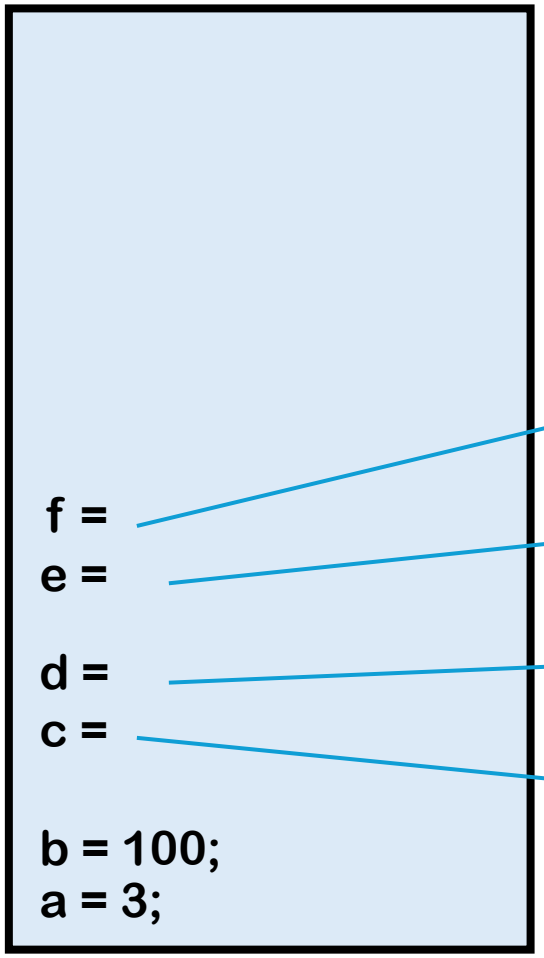
Memory Allocation

GONZAGA UNIVERSITY

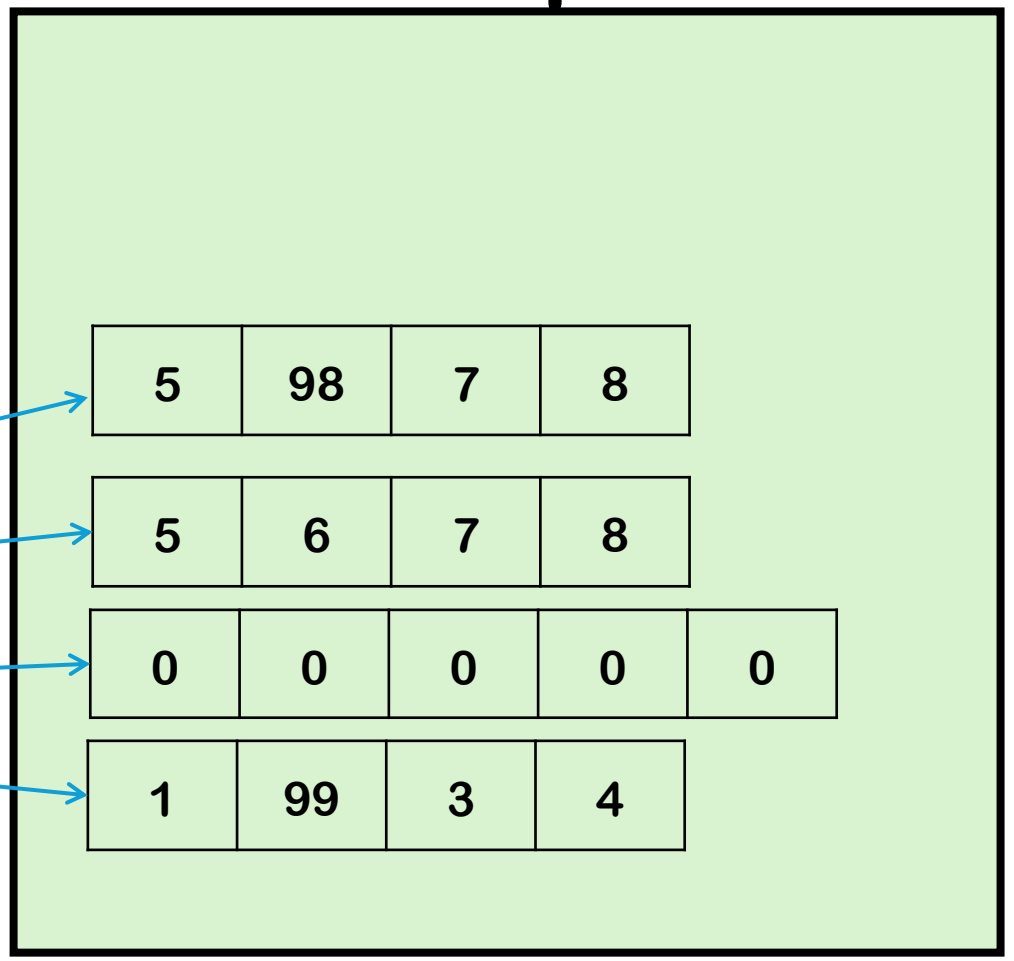
Code

```
int a = 3;
int b = a;
b = 100;
Int [] c = {1,2,3,4};
Int [] d = c;
d[1] = 99;
d = new int [5];
Int [] e = {5,6,7,8};
Int [] f = {5,6,7,8};
f[1] = 98;
```

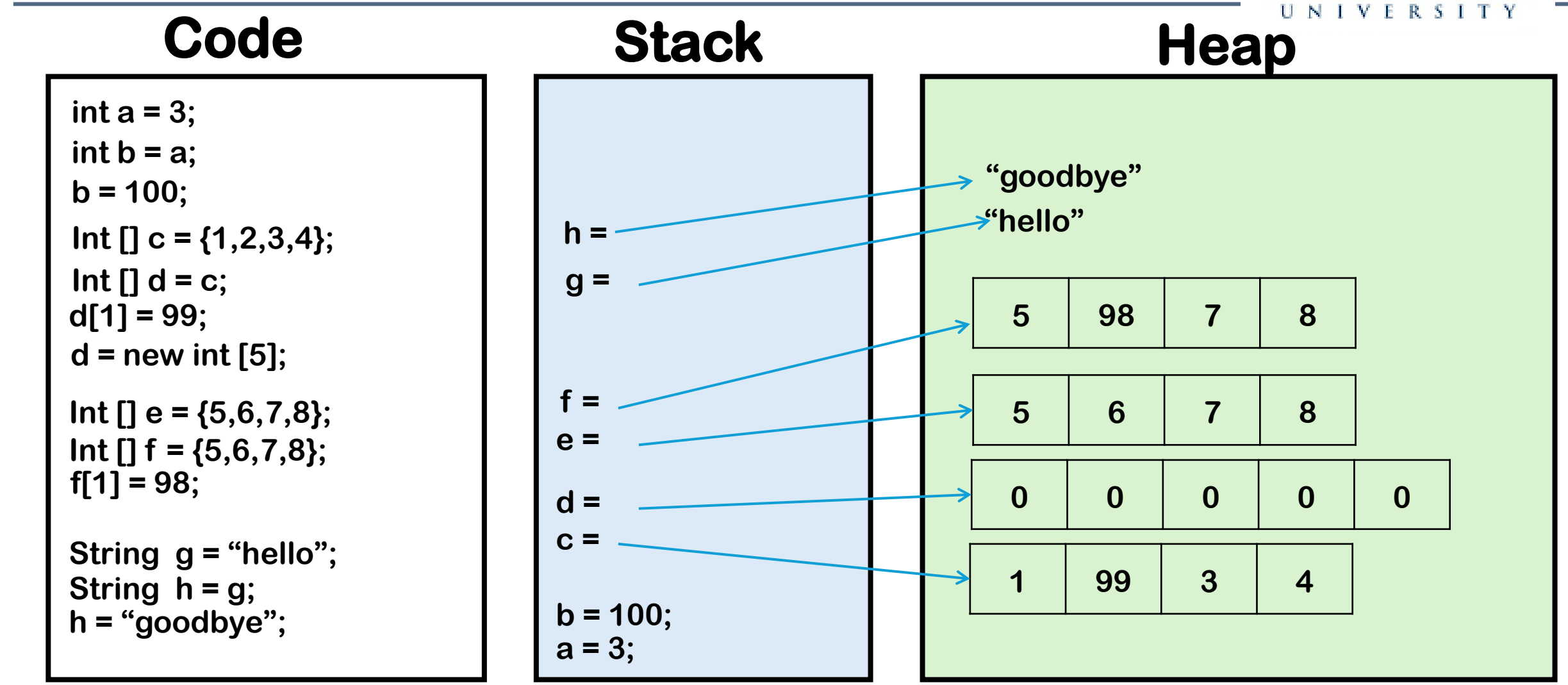
Stack



Heap



Memory Allocation

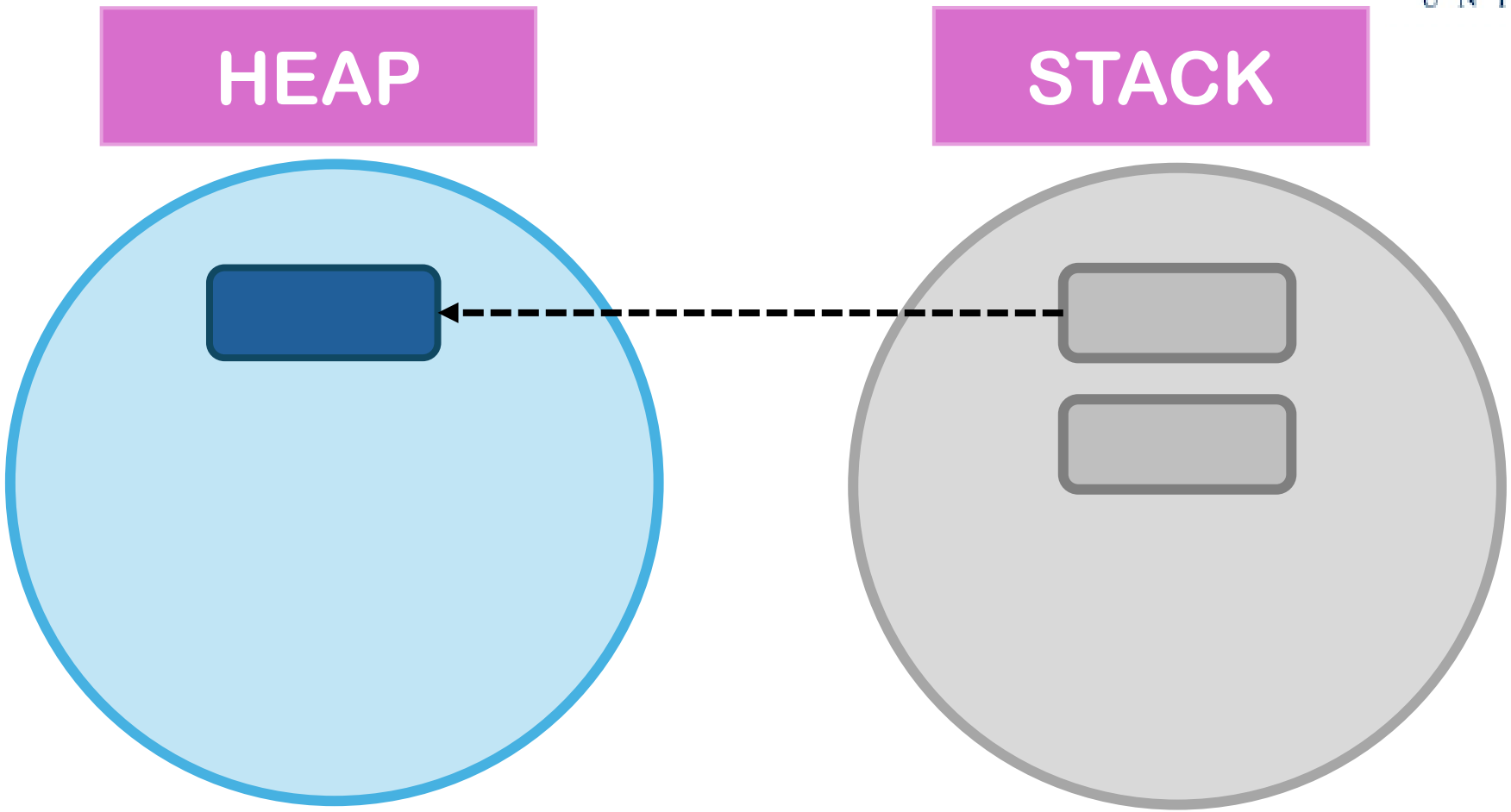


Relationship Between Pointers and Stack

GONZAGA UNIVERSITY

- Pointers can store stack addresses.
- Stack variables are automatically deallocated, so using a pointer to a stack variable after function return leads to dangling pointers.

Memory Allocation Summary



Any Questions?