Welcome to CS / ENGRD 2110!

pollev.com
username: javabear

Get started by signing into PollEv and answering some questions about yourself!

Use your Cornell email address
(but don't "Sign in with Google")

Lecture 1: Introduction

CS 2110

August 25, 2026

Matt Eichhorn

- PhD (Applied Math) at Cornell
- BS (Computer Science, Math)

Teaching:
- CS 2110, CS 2800, ENGRI 1101

Research:
- Algorithm Design: optimization with unknown / random inputs
- Statistics on networks: how do interventions cascade through populations

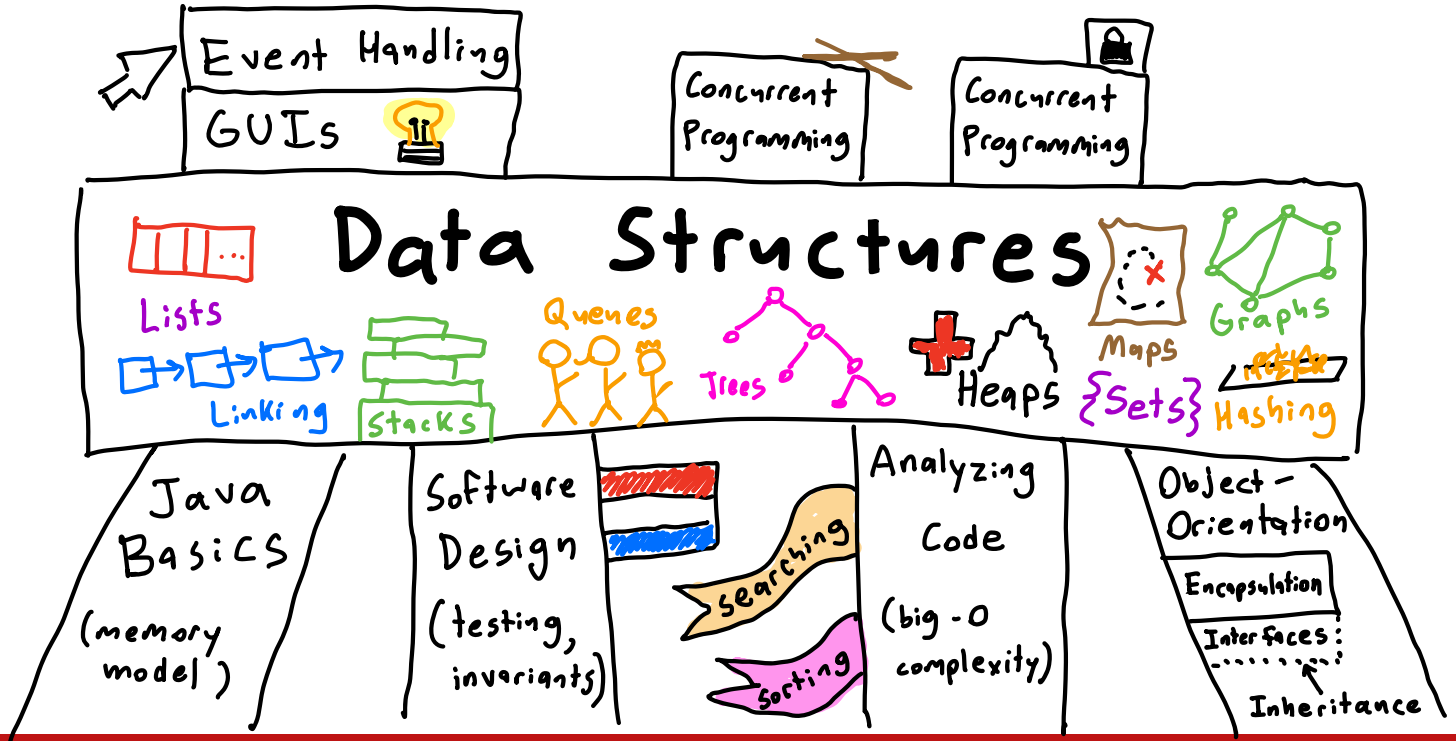
Interests:
- Crosswords / other puzzles, origami, Buffalo Bills

Goal of CS 2110: Write *Better* Code

What makes code *better*?

- more efficient (speed, memory, hardware, energy usage...)
  - > time and space complexity
- easier to read, adapt, maintain, extend (modularity)
  - > future-proof
- user-friendly, helpful features
- reliable, few to no bugs
- robust to security vulnerabilities, smart access control
- ⋮

Course Map


Event Handling
GUIs
Concurrent Programming
Concurrent Programming
Data Structures
Lists
Linking
Stacks
Queues
Trees
Heaps
Maps
{Sets}
Graphs
Hashing
Java Basics
(memory model)
Software Design
(testing, invariants)
searching
sorting
Analyzing Code
(big-O complexity)
Object-Orientation
Encapsulation
Interfaces:
Inheritance

Course Logistics

Everything you need is linked from our course website

courses.cis.cornell.edu/courses/cs2110/2026fa/

(find a link on our Canvas page)

- Syllabus, installation/style guides, etc.
- Lecture Notes (with interactive exercises)
- Slides and demo code (after lecture)
- Discussion (mostly after sections)
- Assignments
- Grades

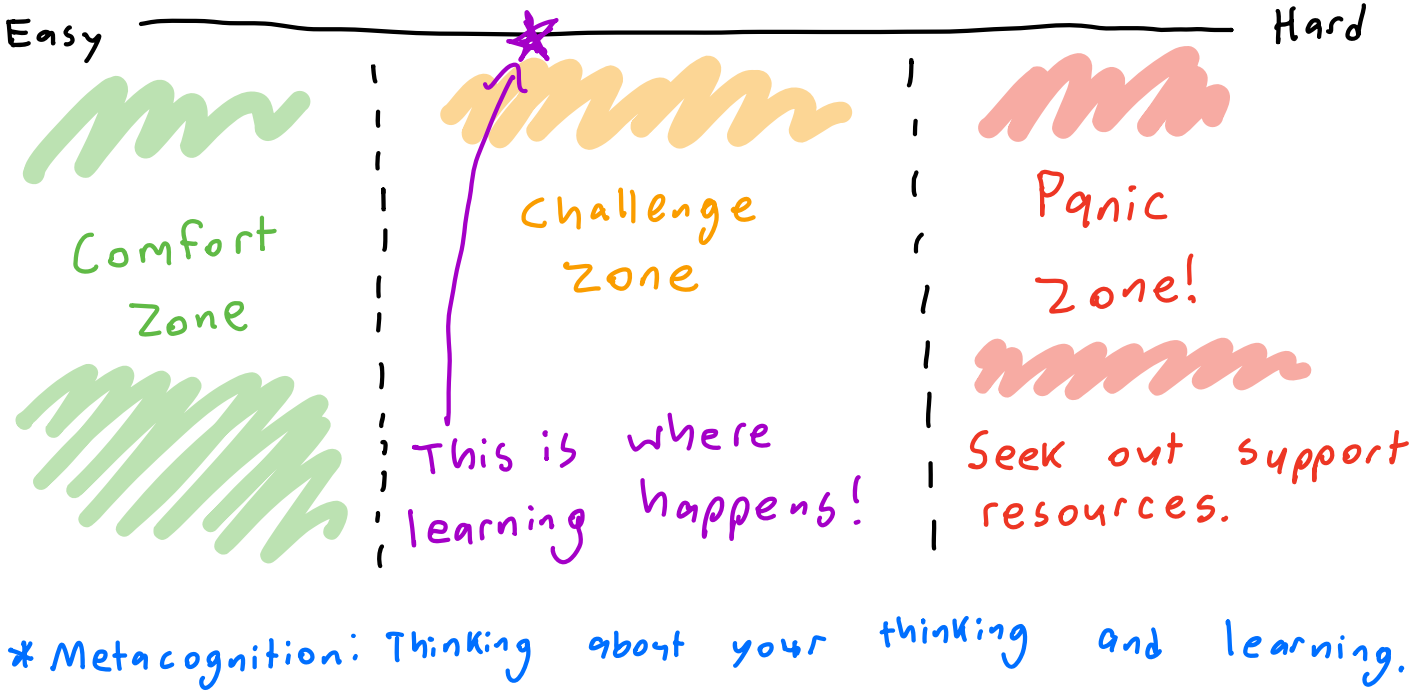
Class Norms

Prioritizing Effort and Integrity:
Assignments are designed for their learning benefit, not their end product
Be pragmatic and transparent with generative AI use
Adhere to the course collaboration policy (syllabus)

Mutual Respect:
We all are here with the same goals
Communicate kindly/supportively to peers and course staff
Let us know if anyone behaves disrespectfully

Inclusivity:
We come with different backgrounds and needs
Everyone in the course is meant to be here
Let the course staff know how we can best support you

Learn by Challenging Yourself



Let the learning begin!

Today's Learning Outcomes

1. Explain the process of compilation and the benefits of *statically typed* programming languages.

2. Determine the static type of an expression involving one of more operators, method calls, and/or implicit/explicit type coercions.

3. Visualize the state of the runtime stack in a program involving multiple method calls.

Poll Everywhere

PollEv.com/javabear text `javabear` to 22333

What is your "go to" programming language?

Java	25% **(A)**	Python	61% **(B)**
C / C++	9% **(C)**	Matlab	0% **(D)**
Javascript / Typescript	2% **(E)**	R	1% **(F)**
Something Else	1% **(G)**		

A Basic Python Program

convert.py

```
1  def days_to_minutes(days):
2    hours = 24 * days
3    minutes = 60 * hours
4    return minutes
5
6  if __name__ == "__main__":
7    print("Enter a number of days: ", end="")
8    days = input() # get user input from the console
9    mins = days_to_minutes(days)
10   print("There are " + mins + " minutes in " + days + " days.")
```

start here

If we run this program and type "3" into the console, what will be its output?

Poll Everywhere

PollEv.com/javabear text `javabear` to 22333

If we run this Python program and type "3" into the console, what will be its output?

There are 1440 minutes in 3 days. **(A)**

There are 4320 minutes in 3 days. **(B)**

(The program will crash because of an error) **(C)**

Something else... **(D)**

A Basic Python Program

convert.py

```
1  def days_to_minutes(days):
2    hours = 24 * days
3    minutes = 60 * hours
4    return minutes
5
6  if __name__ == "__main__":
7    print("Enter a number of days: ", end="")
8    days = input() # get user input from the console
9    mins = days_to_minutes(days)
10   print("There are " + mins + " minutes in " + days + " days.")
```

expecting a number

repeated string concatenation, not multiplication

days is a string, not a number.

Dynamic vs. Static Typing

The **type** of a variable (or expression) determines its possible values and how it can be used within a program.

Dynamically Typed Languages: Infer variable types at runtime
(Python)
- shorter code, but less "safe"

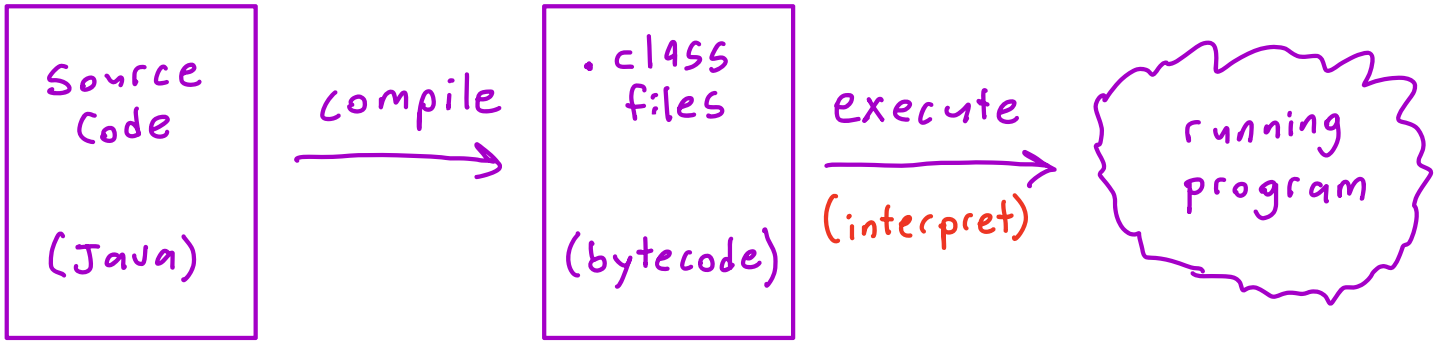
Statically Typed Languages: Types are checked before code is run
(Java)
* often requires explicit type declarations by programmer

Coding Demo: Translation to Java

(We include slides like this one in case you want a place to take notes during/about the demo.)

Compilation

= Intermediary "code translation" step that happens before code runs



During the compilation, the compiler checks types (and other things) and performs optimizations on your code.

Expressions

= units of code with an assigned **type** and **value**

literals: 27 (int) 20.5 (double) true (boolean) 'a' (char)

variables: int age = 27; age (int, value = 27)

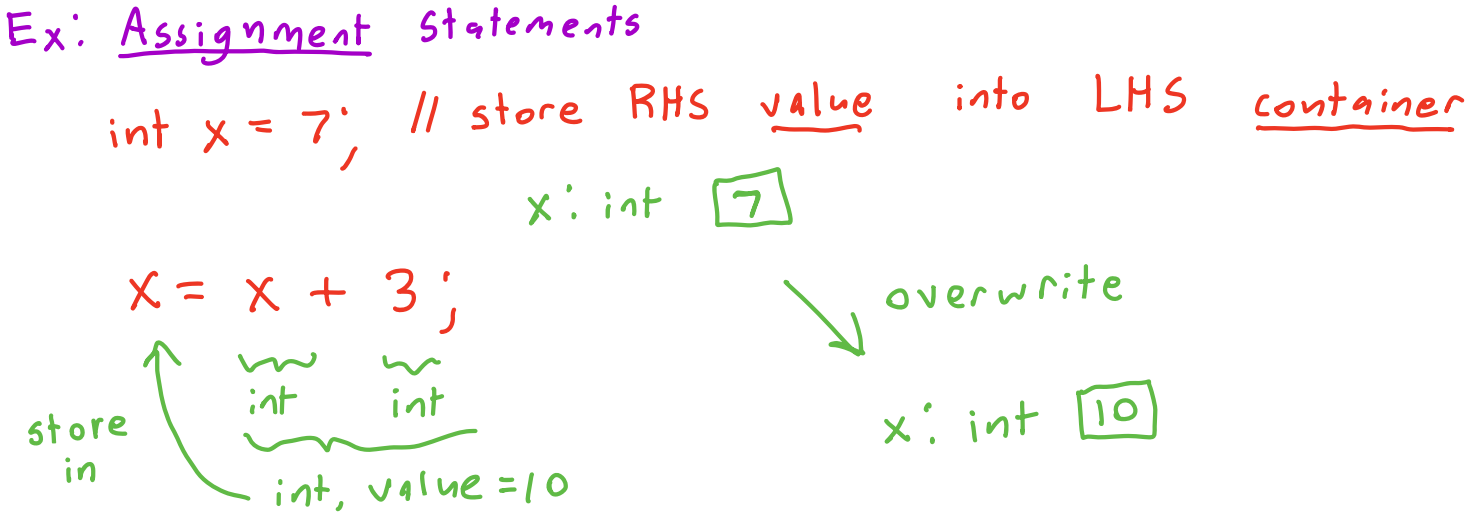
using operators: 3 * 5 (int, value = 15) age + 8 (int, value = 35)

method calls: static int daysToMins(int days) daysToMins(3) (int, value = 4320)

Compiler determines types of all expressions and checks for agreement.

Statements and Assignment

A **statement** is a unit of code that describes an action the program should take (i.e., it produces a **side effect**).



Primitive Type Coercion

Most built-in java operators preserve primitive types.

3 + 4 → 7 (int, int, int)

3.0 + 4.0 → 7.0 (double, double, double)

Coercion alters the type of an expression. It comes in two forms:

Implicit (**widening**): 3 + 4.0 → 3.0 + 4.0 → 7.0 (int, double; double, double; double)

* coercion affects expression evaluation, but doesn't change underlying variables

```
int x = 2;
double y = x;
```

x: int 2

y: double 2.0

Explicit (**casting**): 7 / 2 → 3 (truncated division) (int, int, int)

(double) 7 / 2 → 7.0 / 2 → 7.0 / 2.0 → 3.5

(double) 7: double, value = 7.0

7.0 / 2: widening

Poll Everywhere

PollEv.com/javabear text **javabear** to 22333

What is the (static) type of the highlighted expression?

```
1  int x = 6;
2  boolean y = true;
3  double z = 4.3;
4  System.out.print(x > 2 * z && y);
```

*We didn't get to this in class, but I left it here if you'd like to attempt it. Answer on next slide.

int	(A)
boolean	(B)
double	(C)
String	(D)

Poll Everywhere

PollEv.com/javabear text **javabear** to 22333

What is the (static) type of the highlighted expression?

```
1  int x = 6;
2  boolean y = true;
3  double z = 4.3;
4  System.out.print(x > 2 * z && y);
```

Compiler does a lot of work to figure out types!

int int double boolean

double

boolean

boolean

int	(A)
boolean	(B)
double	(C)
String	(D)

Program Execution

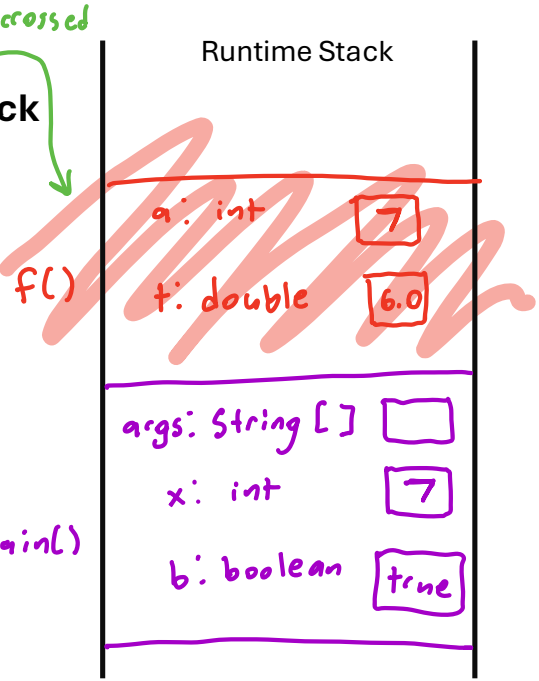
- Starts by calling the `main()` method
- Calling a method creates a **call frame** on the **runtime stack**
- The call frame has an entry (variable box) for **every** parameter/local variable in the method.
- Inner method calls "stack up" call frames
- Call frames are destroyed when methods return

don't typically draw crossed out, just on this slide

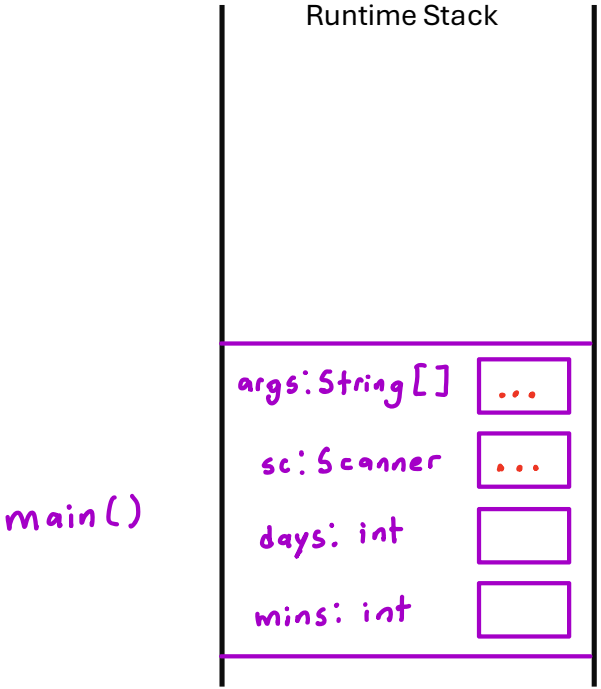
Ex.

```
... main(String[] args){
    int x=7;
    boolean b=f(x);
}

boolean f(int a){
    double t=6.0;
    return a>t;
}
```



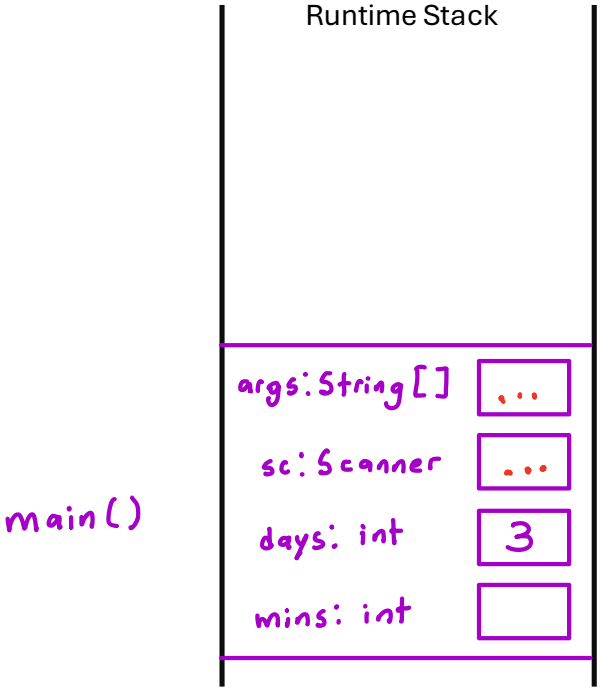
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

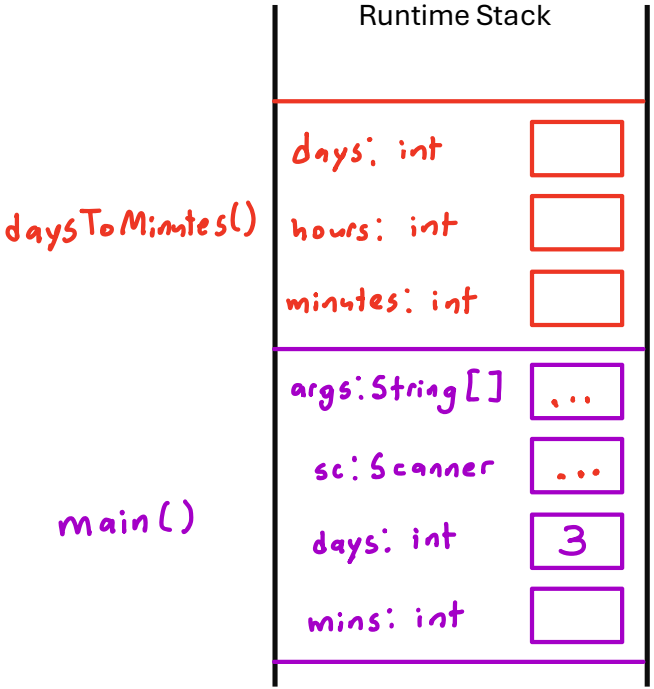
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

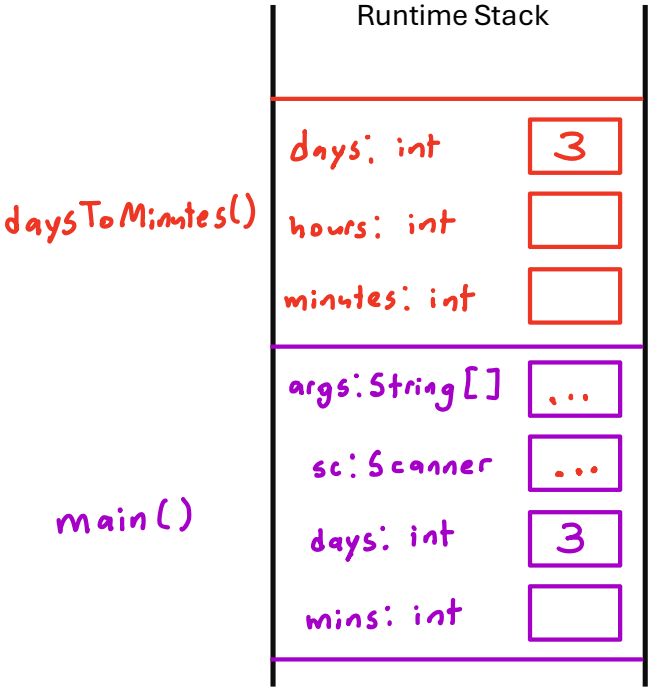
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

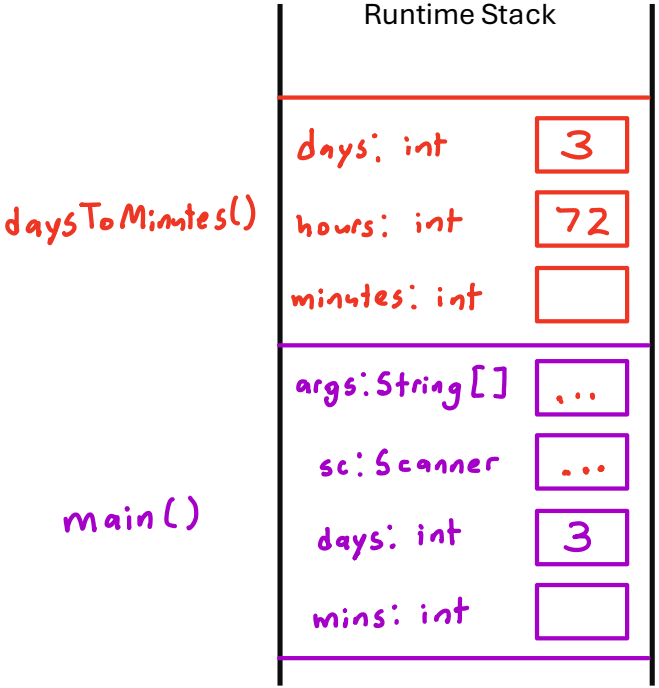
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

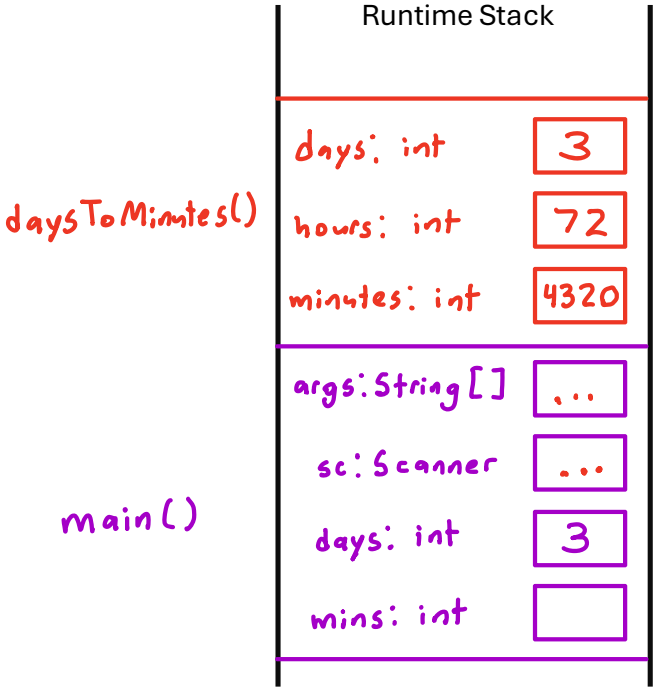
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

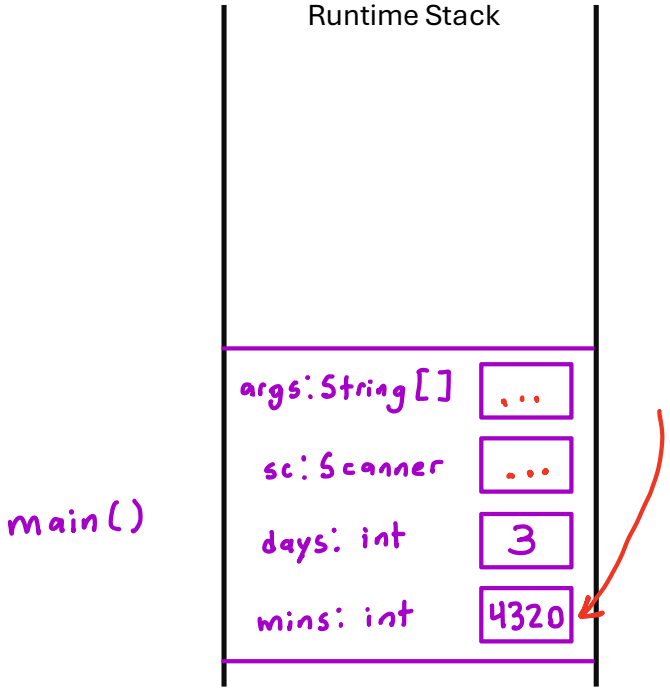
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

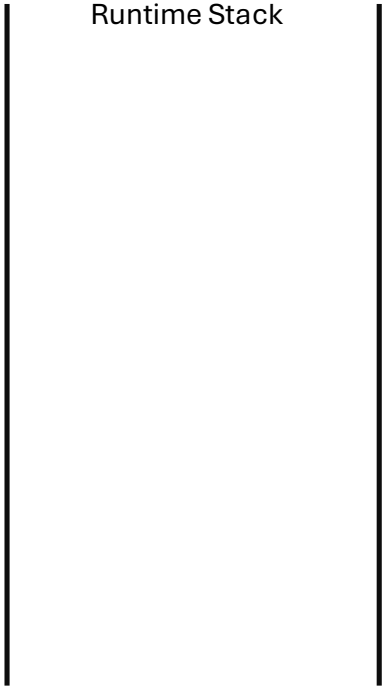
Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```

Diagramming our Code



Convert.java

```
1  public class Convert {
2    static int daysToMinutes(int days) {
3      int hours = 24 * days;
4      int minutes = 60 * hours;
5      return minutes;
6    }
7
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in));
10     int days = sc.nextInt();
11     int mins = daysToMinutes(days);
12     System.out.print(…);
13   }
14 }
```