Lesson 2-1:
Variables, Objects, and Methods

Computer Science 46A: Introduction to Programming

San José State University

Announcements

- Watch Canvas for Homework
- Lab #2 is this Friday (2/7)
- Zybooks should now be integrated
    - Catch up required
    - Reading assignments before class
        - Roughly 1 hour each week
        - ~1 chapter every week
        - Graded
    - Challenge assignments due end of week
        - Also in zybooks

Learning Objectives

By the end of this lesson, you should be able to:

1. Declare variables and objects within a class's `main` method
2. Describe the components of constructors and methods
3. Utilize constructors and methods from an existing class

Variables

Declaring Variables

- Variables are used to carry data through a program
- In Java, variables must be explicitly declared before they are used
    - This is not always the case – in Python, for example, this is taken care of implicitly (you don't have to declare a variable before you use it!)
- Variables are only declared once (otherwise you will get a syntax error)
- Example:

`SumAndProductOfEvenNumbers.java (revised)`

Some Primitive Data Types in Java

Data Type	Description	Example
int	A number with no decimal places	15
double	A number that may have up to 15 digits	2.7182818284
char	A single character (uses single quotes ‘ ‘)	‘a’
boolean	A true or false variable	true or false

- To declare a primitive variable, we use the following syntax:

```
    [type] [var name];
```

- A primitive variable may be declared and a value may be assigned to it at the same time:

```
    [type] [var name] = [value];
```

Declaration Example: `double`

- A double variable called piValue may be declared with

```
double piValue;
```

- And then assigned the value 3.1415 with

```
piValue = 3.1415;
```

- Or it can be done in one line:

```
double piValue = 3.1415;
```

Some Non-Primitive Data Types in Java

Data Type	Description	Example
String	A list of characters (uses double quotes " ")	"Hey Everyone!"
Class	A set of objects and associated methods	Picture

When a variable refers to an object that uses a constructor, we use the `new` operator:

```
[Class] [obj name] = new [Class](inputs);
```

*The new operator should not be used for Strings

Question:
What is the output of the following code?

```
int myFavoriteNumber = 7;
int myFriendsFavoriteNumber;
myFriendsFavoriteNumber = myFavoriteNumber + 1;

System.out.print(myFavoriteNumber + “, ”);
System.out.println(myFriendsFavoriteNumber);
```

A) 7, 7

B) 7, 8

C) 8, 8

D) Compile Error

Style Alert: Meaningful Names

When naming a variable, give it a name that describes what it is

For example,

```
int myHeightInches = 70;
```
😀

is much better than

```
int mhi = 70;
```
😡

When I access the variable `myHeihgtInches` later in the program, I’ll remember what it is – not the case for `mhi`

This is especially important when you are sharing code with other people!

Style Alert: camelCase for variables

- In coding, its best to stick with one convention for formatting variable names other components of your code
- In Java, our convention for formatting variable names will be

  ”Camel Case”

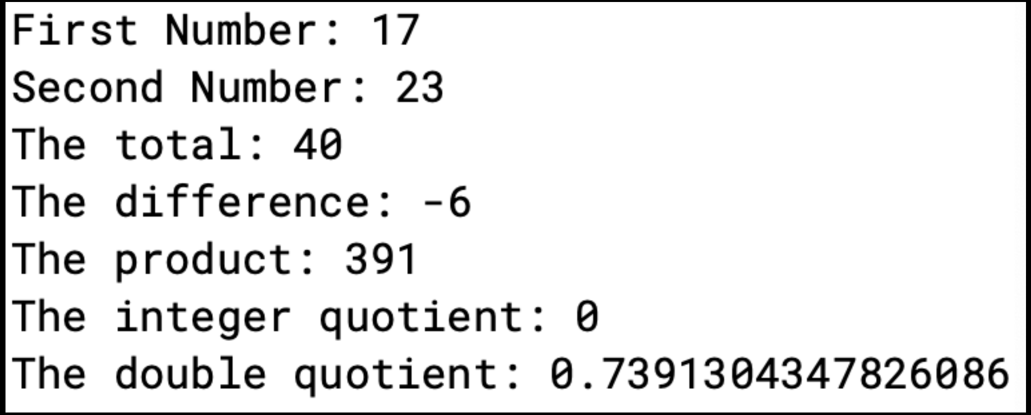
- Camel Case removes spaces and capitalizes every word except the first, e.g.

```
aGoodVariableName     😃
Abad_variablename     😡
```

In other languages or environments, the people you’re working with may prefer other styles such as “Pascal Case” or “Snake Case”

Participation Exercise 2-1a: `VariablesAndAssignment`

<u>Goal:</u> Declare variables and perform some simple arithmetic



```
First Number: 17
Second Number: 23
The total: 40
The difference: -6
The product: 391
The integer quotient: 0
The double quotient: 0.7391304347826086
```

Output of `VariablesAndAssignment`

Note: the symbols for adding, subtracting, multiplying and dividing are:

`+, -, *, /`

Objects Revisited

Review: How are objects used?

First, they are created

```
   Picture pic = new Picture("dogcat.png");
```

Then, methods are called on the objects:

```
   pic.draw();
   pic.grow(25, -10);
   pic.translate(20, -10);
```

So far, we've created objects using the Picture class and called some methods without fussing too much about how these are created

Let's take a look under the hood to see how they are constructed

Object Constructors in a Class

- Objects are created in a class using <u>constructors</u> with the same name as the class, e.g

```
public [class name](inputs)
```

- There may be several different constructors with different input options
- For example, in the Picture class, there are three constructors:

```
public Picture()
public Picture(double width, double height)
public Picture(String source)
```

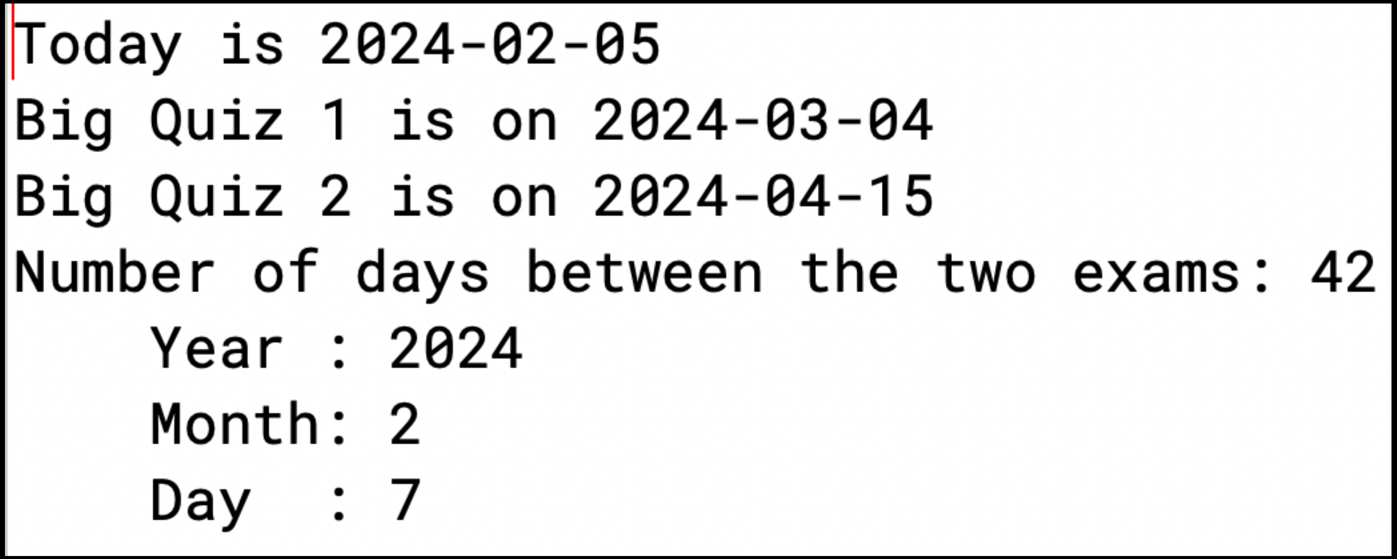
arguments are listed by their type and then their name

Example: The Day Class

- In this lesson we will use the Day class
- This class is provided in the Lesson02-1.zip file
- Open par02-1b, open the BlueJ project, open the Day class in the editor

Participation Exercise 2-1b: DayProg

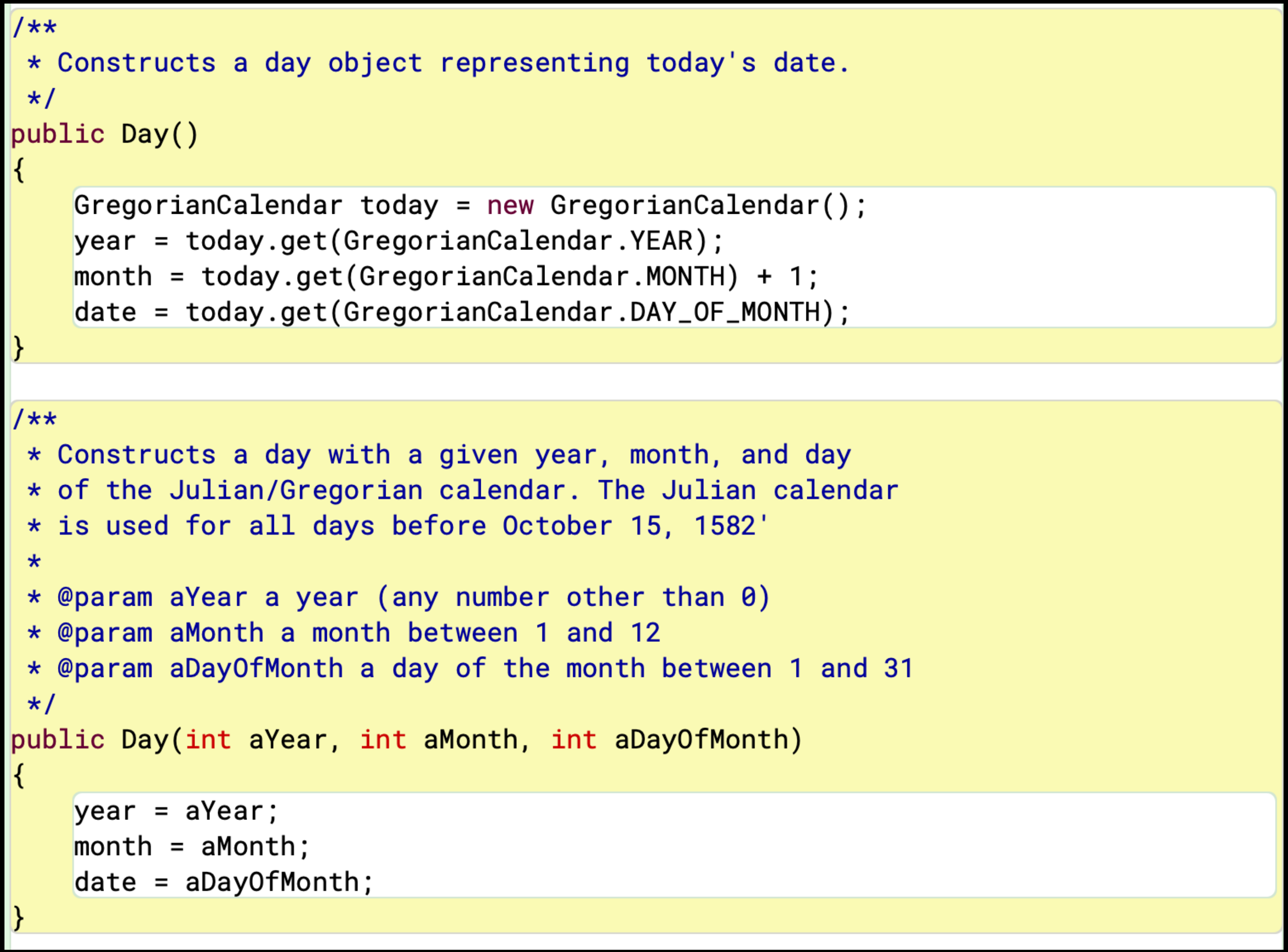
Goal: Use methods of the Day class to print the dates of quizzes and exams in this course



```
Today is 2024-02-05
Big Quiz 1 is on 2024-03-04
Big Quiz 2 is on 2024-04-15
Number of days between the two exams: 42
    Year : 2024
    Month: 2
    Day  : 7
```

Output of DayProg

Constructors in the Day Class



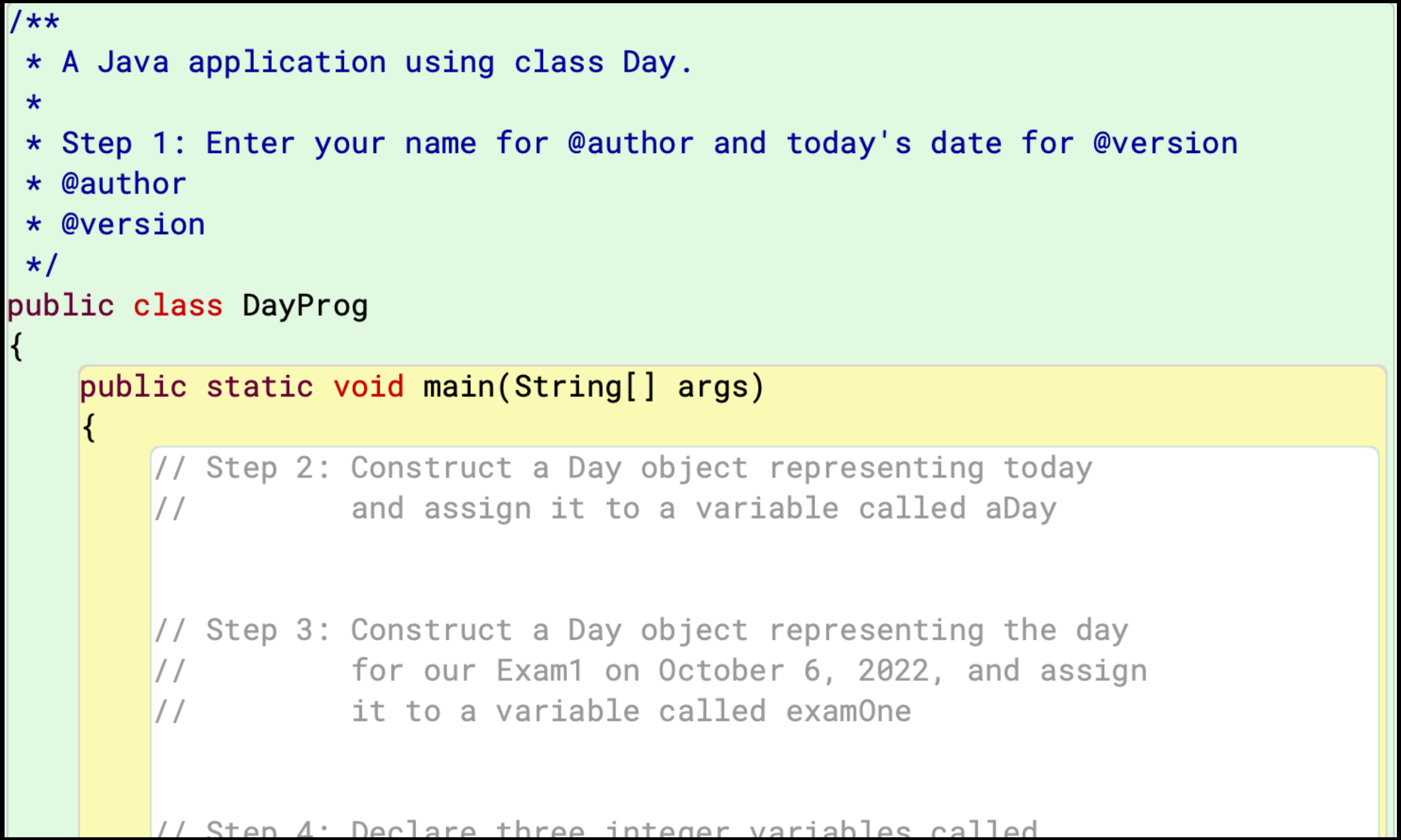
```java
/**
 * Constructs a day object representing today's date.
 */
public Day()
{
    GregorianCalendar today = new GregorianCalendar();
    year = today.get(GregorianCalendar.YEAR);
    month = today.get(GregorianCalendar.MONTH) + 1;
    date = today.get(GregorianCalendar.DAY_OF_MONTH);
}

/**
 * Constructs a day with a given year, month, and day
 * of the Julian/Gregorian calendar. The Julian calendar
 * is used for all days before October 15, 1582'
 *
 * @param aYear a year (any number other than 0)
 * @param aMonth a month between 1 and 12
 * @param aDayOfMonth a day of the month between 1 and 31
 */
public Day(int aYear, int aMonth, int aDayOfMonth)
{
    year = aYear;
    month = aMonth;
    date = aDayOfMonth;
}
```

There are 2 constructors in the day class

The first constructor takes no arguments and will produce an object which stores the year, month, and day of today's date

The second constructor take 3 arguments and will produce an object which stores the given year, month, and day

Steps 2-3 in DayProg



```
/**
 * A Java application using class Day.
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author
 * @version
 */
public class DayProg
{
    public static void main(String[] args)
    {
        // Step 2: Construct a Day object representing today
        //         and assign it to a variable called aDay


        // Step 3: Construct a Day object representing the day
        //         for our Exam1 on October 6, 2022, and assign
        //         it to a variable called examOne


        // Step 4: Declare three integer variables called
```

Recall that objects are declared as

```
[class] [obj name] = new [Class](inputs);
```

Method Declarations in a Class

- Methods which can be called on an object are declared in a class using the following syntax:

```
    public [output type] [method name](inputs)
```

- For example, the Picture class had the following methods:

```
    public int getHeight()
    public void grow(double dw, double dh)
```

when a method does not return a value, its output type is declared as void

arguments are listed by their type and then their name

Methods in the Day Class

The Day class has the following public methods:

- `getYear()`
- `getMonth()`
- `getDayOfMonth()`
- `addDays()`
- `daysFrom()`
- `toString()`

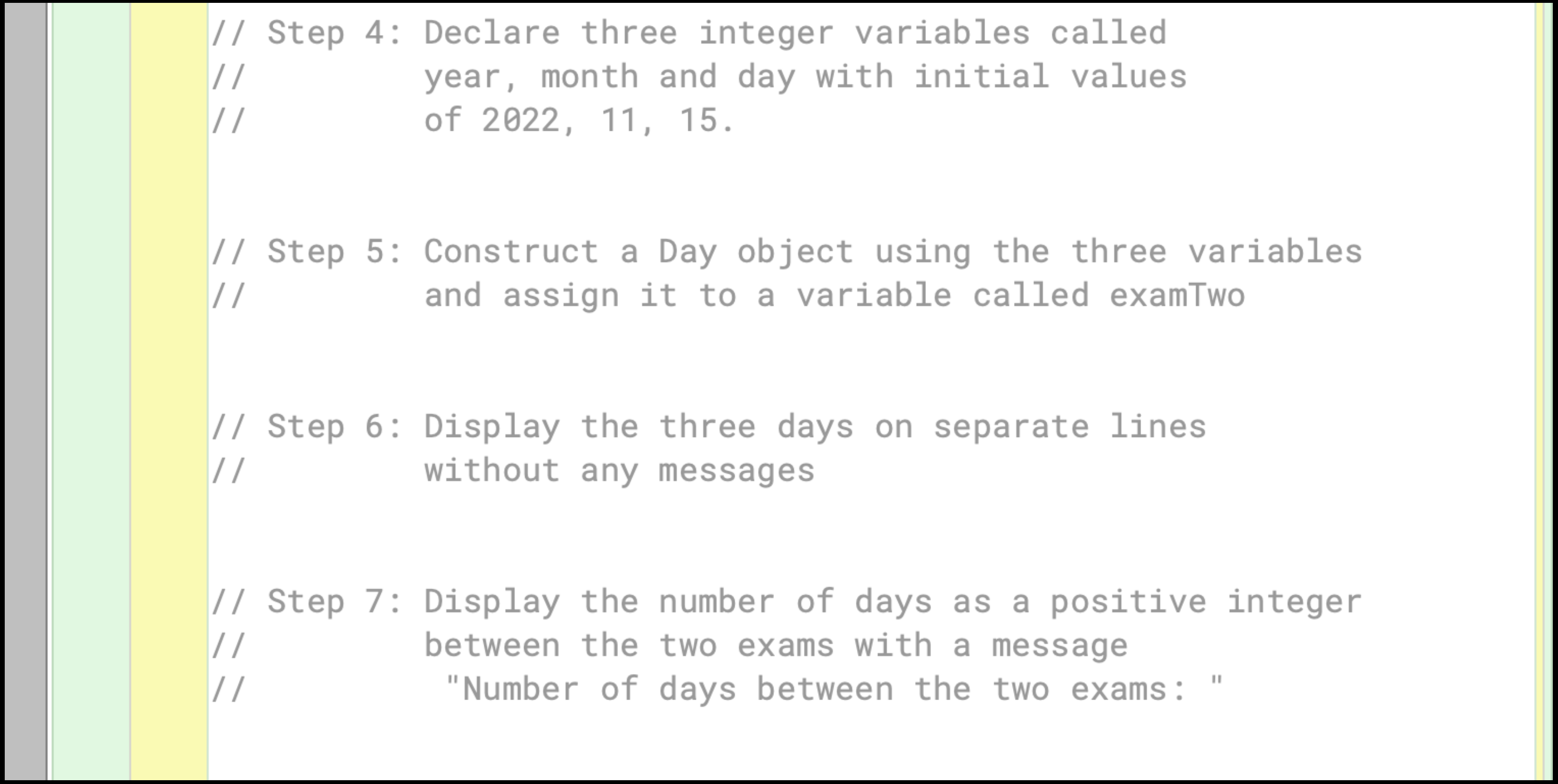
Public methods can be called outside of a class

The Day class has the following private methods:

- `compareTo()`
- `nextDay()`
- `previousDay()`
- `daysPerMonth()`
- `isLeapYear()`

Private methods can only be called within a class

Steps 4-7 in DayProg



```
// Step 4: Declare three integer variables called
//         year, month and day with initial values
//         of 2022, 11, 15.


// Step 5: Construct a Day object using the three variables
//         and assign it to a variable called examTwo


// Step 6: Display the three days on separate lines
//         without any messages


// Step 7: Display the number of days as a positive integer
//         between the two exams with a message
//          "Number of days between the two exams: "
```

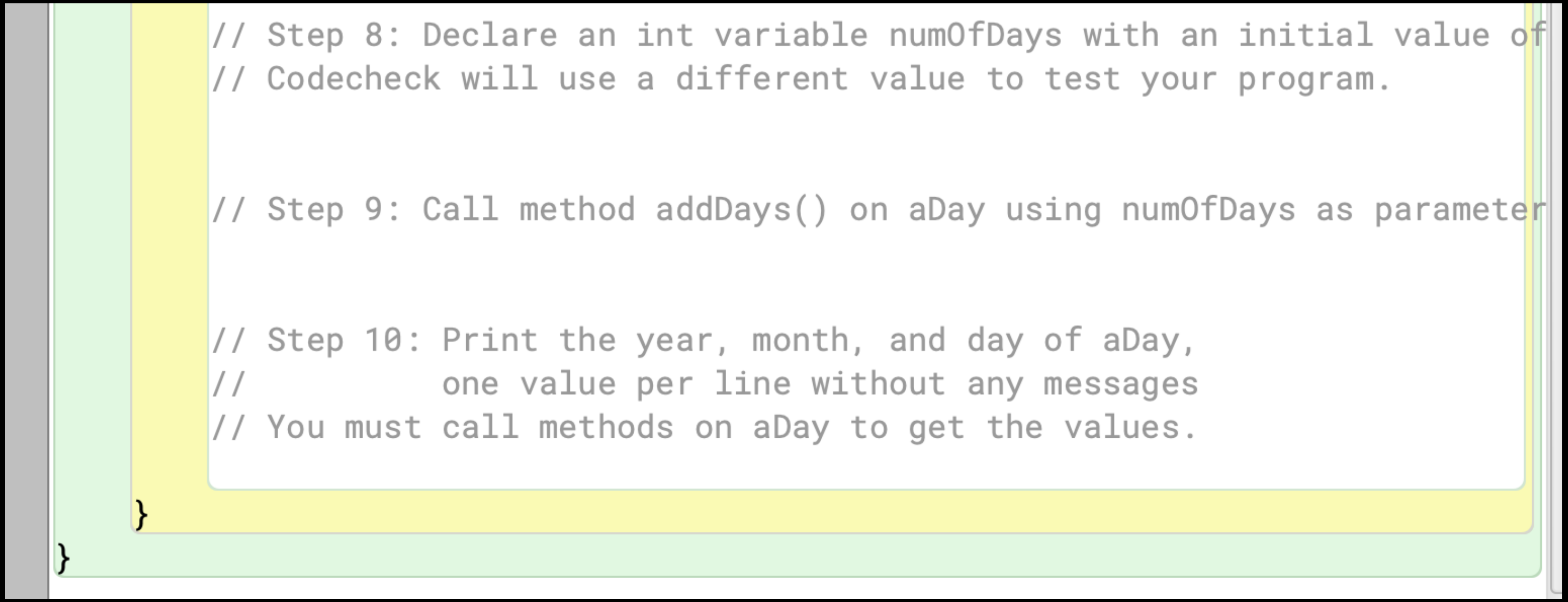
Recall that variables are declared and defined as

`[type] [var name] = [value];`

Method Types

- Methods can have two types of behavior relative to the object
- Accessors only access data about the object without changing it
    - E.g. from the Picture class, we used the .getX() method which just returned some information about the pic object
- Mutators change an object's data
    - E.g. from the Picture class, we used the .grow() method which altered the size of the image

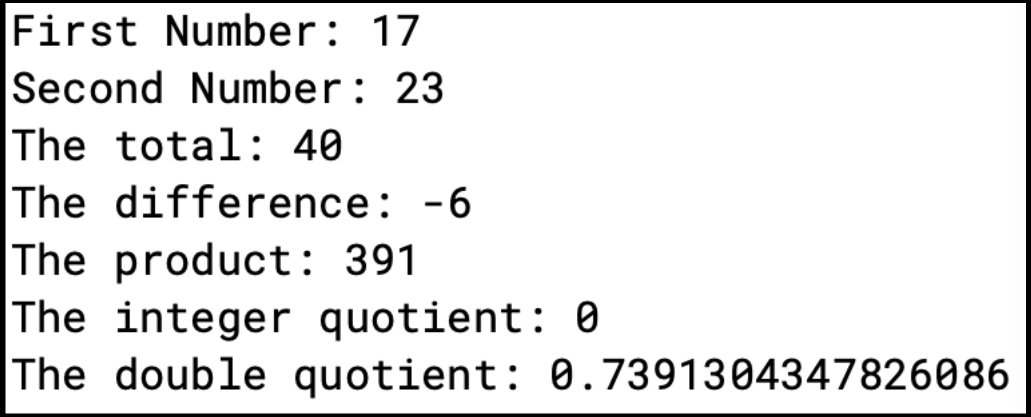
Steps 8-10 in DayProg



```
        // Step 8: Declare an int variable numOfDays with an initial value of
        // Codecheck will use a different value to test your program.


        // Step 9: Call method addDays() on aDay using numOfDays as parameter


        // Step 10: Print the year, month, and day of aDay,
        //          one value per line without any messages
        // You must call methods on aDay to get the values.

    }
}
```

Submit Participation Exercise 2-1a: `VariablesAndAssignment`

<u>Goal:</u> Declare variables and perform some simple arithmetic



```
First Number: 17
Second Number: 23
The total: 40
The difference: -6
The product: 391
The integer quotient: 0
The double quotient: 0.7391304347826086
```

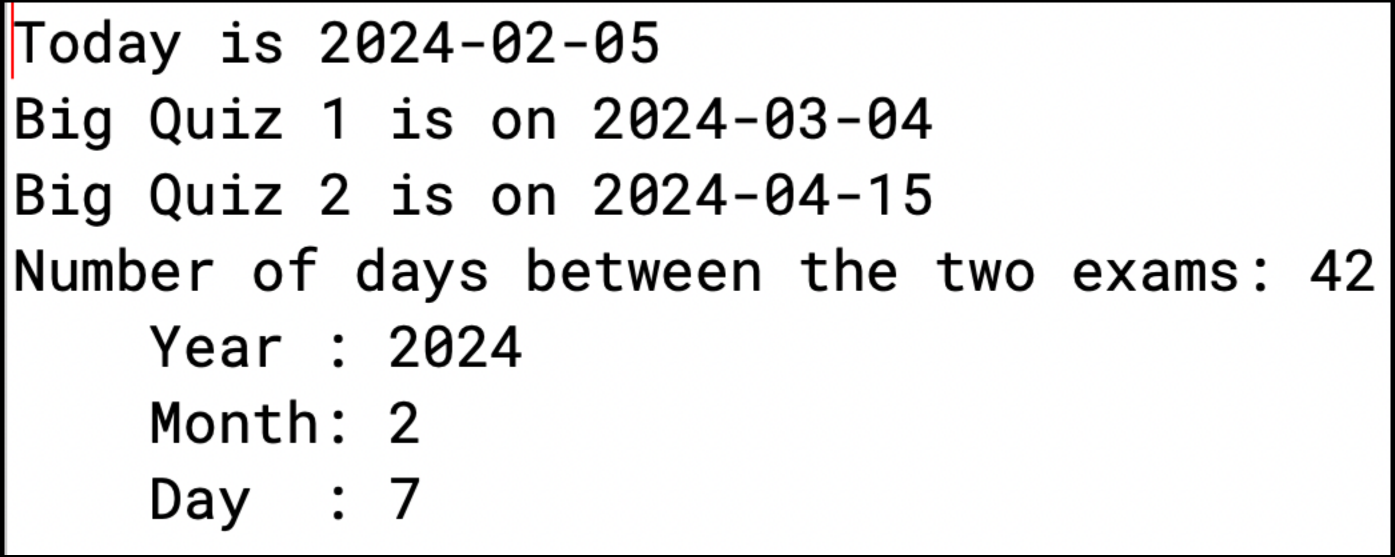
Output of `VariablesAndAssignment`

Note: the symbols for adding, subtracting, multiplying and dividing are:

`+, -, *, /`

Submit Participation Exercise 2-1b: DayProg

Goal: Use methods of the Day class to print the dates of quizzes and exams in this course



```
Today is 2024-02-05
Big Quiz 1 is on 2024-03-04
Big Quiz 2 is on 2024-04-15
Number of days between the two exams: 42
    Year : 2024
    Month: 2
    Day  : 7
```

Output of DayProg