Unit 1: Primitive Types
Variables and Datatypes

Adapted from:

1) Building Java Programs: A Back to Basics Approach
by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum

Textbook Reference

Online Textbook Think Java - 2nd Edition by Allen Downey and Chris Mayfield

For this lecture use Chapter 2

Practice using Trinket

- **Practice all of the examples in this unit using a blank Java Trinket or a specific one using extracts provided via Slack**
- **Name the Java Class and Trinket per instructions.**
- **Run them, fix errors. Submit each of them when complete**
- **Do this daily during class and for homework afterward**

Data Types

A **type** is a set of values (e.g. integers, floats, etc..) and a set of operations (e.g. +, -, *, /, etc..) on them.

Data types can be categorized as either **primitive** or **reference**.

The primitive data types used in this course define the set of operations for numbers and Boolean(true or false) values.

Reference variables or object variables hold a reference(or address) to an object of a class(more on this later).

Primitive types

The primitive types on the Advanced Placement Computer Science A exam are:

- **int** - which store integers (whole numbers like 3, -76, 20393)
- **double** - which store floating point numbers (decimal numbers like 6.3, -0.9, and 60293.93032)
- **boolean** - which store Boolean values (either true or false).

Receipt example

What's bad about the following code?

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);
        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);
        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);
        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .08 +
                           (38 + 40 + 30) * .15);
    }
}
```

- The subtotal expression `(38 + 40 + 30)` is repeated
- So many `println` statements
- We will use **variables** to solve the above problems.

Variables

- **variable**: A piece of the computer's memory that is given a name and type, and can store a value.
  - Like preset stations on a car stereo, or cell phone speed dial:

  - Steps for using a variable:
    - Declare it - state its name and type
    - Initialize it - store a value into it
    - Use it - print it or use it as part of an expression

Declaration

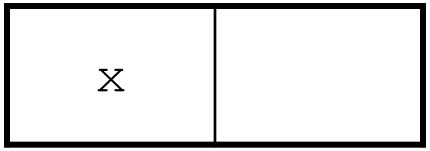
- **variable declaration**: Sets aside memory for storing a value.
  - Variables must be declared before they can be used.

- Syntax:

  type name;

  - The name is an identifier.

  - `int x;`



  - `double myGPA;`

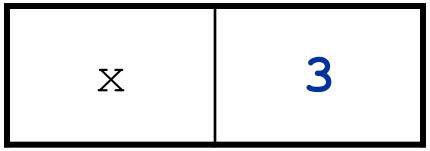
Assignment

- **assignment**: Stores a value into a variable.
  - The value can be an expression; the variable stores its result.

- Syntax:

```
name = expression;
```

  - ```
    int x;
    x = 3;
    ```




  - ```
    double myGPA;
    myGPA = 1.0 + 2.25;
    ```

Using variables

- Once given a value, a variable can be used in expressions:

```
int x;
x = 3;
System.out.println("x is " + x);      // x is 3
System.out.println(5 * x - 1);        // 14
```

string concatenation:
string + number = concatenated string
(more on this later)

- You can assign a value more than once:

```
int x;
x = 3;
System.out.println(x + " here");      //
```


```
x = 4 + 7;
System.out.println("now x is " + x); // now x is 11
```

Declaration/initialization

- A variable can be declared/initialized in one statement.

- Syntax:

```
type name = value;
```

  - `double myGPA = 3.95;`

myGPA	3.95

  - `int x = (12 - 3) * 2;`

x	18

Assignment and algebra

- Assignment uses = , but it is not an algebraic equation.

  = means, "store the value at right in variable at left"

  - The right side expression is evaluated first, and then its result is stored in the variable at left.

- What happens here?

```
int x = 3;
x = x + 2;  // no solutions
            // mathematically
            // not an equation!
```

Multiple Variables

- Multiple variables of the same type can be declared and initialized at the same time.

- Syntax:

type name1, name 2, name3;

type name1 = value1, name2 = value2, name3 = value3;

```
int x, y, z;  // declare three integers.
int a = 1, b = 2, c = 3;// declare and initialize
                        // three integers.
```

Assignment and types

- A variable can only store a value of its own type.
  - `int x = 2.5;` **`// ERROR: incompatible types`**

- An `int` value can be stored in a `double` variable.
  - The value is converted into the equivalent real number.
  - `double myGPA = 4;`

Compiler errors

- Order matters.
  - `int x;`

    `7 = x; // ERROR: should be x = 7;`
- A variable can't be used until it is assigned a value.
  - `int x;`

    `System.out.println(x);  // ERROR: x has no value`
- You may not declare the same variable twice.
  - `int x;`
    `int x;  // ERROR: x already exists`
  - `int x = 3;`
    `int x = 5;  // ERROR: x already exists`
    - How can this code be fixed?

Printing a variable's value

- Use + to print a string and a variable's value on one line.

```
double grade = (95.1 + 71.9 + 82.6) / 3.0;
System.out.println("Your grade was " + grade);

int students = 11 + 17 + 4 + 19 + 14;
System.out.println("There are " + students +
                   " students in the course.");
```

- Output:

```
Your grade was 83.2
There are 65 students in the course.
```

Receipt question

Improve the receipt program using variables.

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);

        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);

        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);

        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .15 +
                           (38 + 40 + 30) * .08);
    }
}
```

Receipt answer

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        int subtotal = 38 + 40 + 30;
        double tax = subtotal * .08;
        double tip = subtotal * .15;
        double total = subtotal + tax + tip;

        System.out.println("Subtotal: " + subtotal);
        System.out.println("Tax: " + tax);
        System.out.println("Tip: " + tip);
        System.out.println("Total: " + total);
    }
}
```

Type `boolean`

- **`boolean`**: A logical type whose values are `true` and `false`.

```
int age = 22;
boolean minor     = (age < 21);
boolean lovesAPCS = true;
System.out.println(minor); // false
System.out.println(lovesAPCS); // true
```

final

- The keyword **final** can be used in front of a variable declaration to make it a constant that cannot be changed. Constants are traditionally capitalized.

```
public class TestFinal
{
   public static void main(String[] args)
   {
     final double PI = 3.14;
     System.out.println(PI);
     PI = 4.2; // This will cause a syntax error
   }
}
```

Naming variables

The name of the variable should describe the data it holds. A name like `score` helps make your code easier to read.

A name like x is not a good variable name in programming, because it gives no clues as to what kind of data it holds.

Do not name your variables crazy things like `thisIsAReallyLongName`, especially on the AP exam. You want to make your code easy to understand, not harder.

Naming variables

The convention in Java and many programming languages is to always start a variable name with a lower case letter and then uppercase the first letter of each additional word.

Variable names **can not include spaces** so uppercasing the first letter of each additional word makes it easier to read the name. Uppercasing the first letter of each additional word is called **camel case**.

```
int numOfLives = 3; // camel case to highlight words
```

Another option is to use underscore symbol _ to separate words, but you cannot have spaces in a variable name. Java is case sensitive so playerScore and playerscore are not the same.

```
int num_of_lives = 3;  // use _ to highlight words.
```

Keywords

- **keyword**: An identifier that you cannot use to name a variable because it already has a reserved meaning in Java.

```
abstract    default     if          private     this
boolean     do          implements  protected   throw
break       double      import      public      throws
byte        else        instanceof  return      transient
case        extends     int         short       try
catch       final       interface   static      void
char        finally     long        strictfp    volatile
class       float       native      super       while
const       for         new         switch
continue    goto        package     synchronized
```

trinket.io assignments

The following labs are trinket.io assignments. Log on to your account to complete them. They are included here for your reference.

Name: Lab001.java

Lab 001: Create Variables and Printing Them:

For you to do:

1. Create two String variables called firstName and lastName respectively.
2. Assign the String "Bob" to firstName and "Jones" to lastName.
3. Print firstName and lastName, one per line. Each should use a different print statement.

trinket.io assignments

The following labs are trinket.io assignments. Log on to your account to complete them. They are included here for your reference.

Name: Lab002.java

Lab 2: Create Variables and Printing 2:

For you to do:

1. Create an integer variable on line 4 and set it to the value 21.

trinket.io assignments

The following labs are trinket.io assignments. Log on to your account to complete them. They are included here for your reference.

Name: Lab003.java

Lab 3: Creating Variables and Printing 3

For you to do:

- Create a boolean variable called "isTrue" and set it to false
- Create a double variable called "money" and set it to 99999.99
- Print the variable "money" first
- Then print the variable "isTrue" **without skipping to**

References

1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum: https://runestone.academy/runestone/books/published/csawesome/index.html

For more tutorials/lecture notes in Java look in

CPJava Website

Runestone CSAwesome BUSHSCHOOL_CPJAVA Course

References

1) CPJava Website
2) CPJava Google Classroom
3) CPJava trinket.io Classroom
4) Runestone CSAwesome BUSHSCHOOL_CPJAVA Course
5) Online Textbook Think Java - 2nd Edition by Allen Downey and Chris Mayfield
6) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp