Data Type and Variables

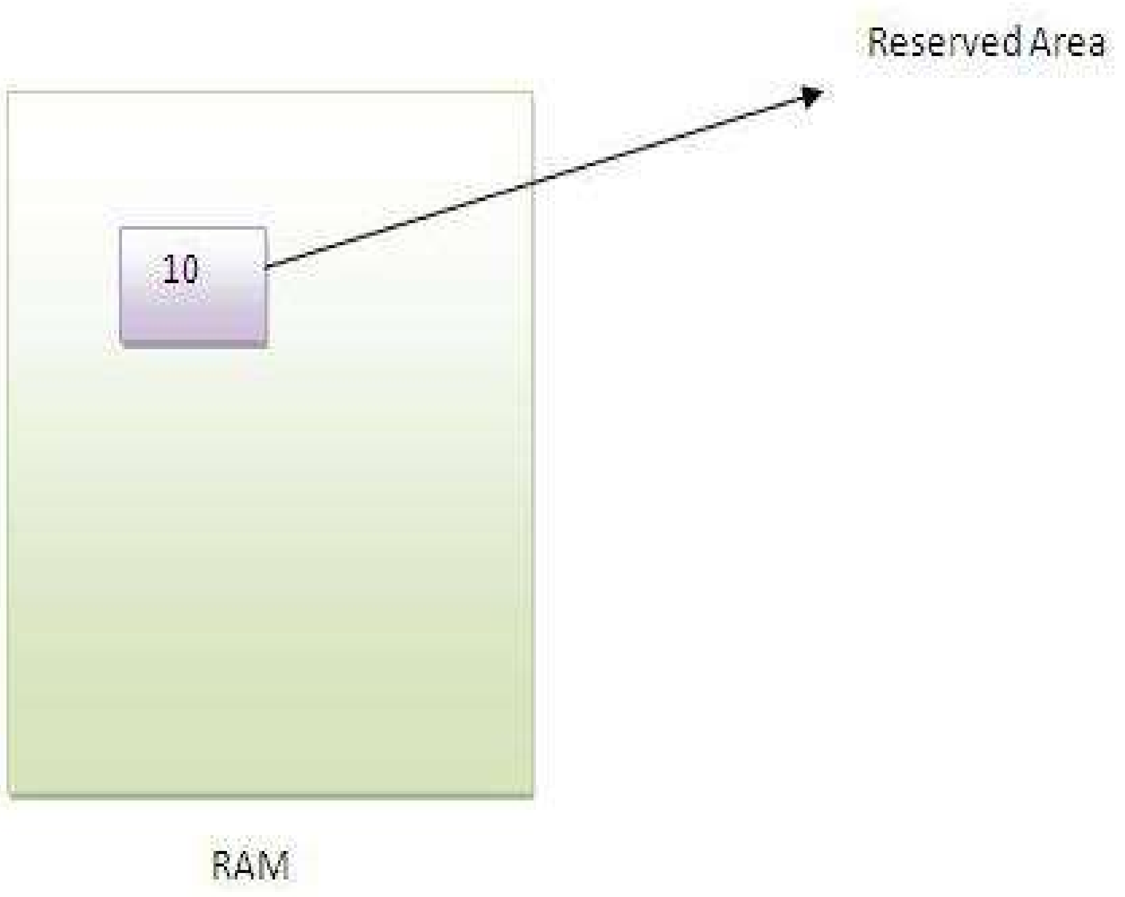
Prepared by

Dr. Rajesh Kumar Ojha

Asst. Prof., CSE, Silicon University

Variables?

- Variable is name of reserved area allocated in memory.



Variables?

- The Java programming language defines the following kinds of variables:
  - Instance Variable
  - Class Variable
  - Local Variable
  - Parameters

Instance Variables

- **Instance Variables (Non-Static Fields)** Technically speaking, objects store their individual states in "non-static fields", that is, fields declared without the static keyword.
- Non-static fields are also known as *instance variables* because their values are unique to each *instance* of a class (to each object, in other words); thecurrentSpeed of one bicycle is independent from the currentSpeed of another.

Class Variables

- **Class Variables (Static Fields)** A *class variable* is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated.
- A field defining the number of gears for a particular kind of bicycle could be marked as static since conceptually the same number of gears will apply to all instances.
- The code static int numGears = 6; would create such a static field. Additionally, the keyword final could be added to indicate that the number of gears will never change.

Local Variables

- **Local Variables** Similar to how an object stores its state in fields, a method will often store its temporary state in *local variables*. The syntax for declaring a local variable is similar to declaring a field (for example, int count = 0;).
- There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared — which is between the opening and closing braces of a method. As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.

Class by Example

- In the real world, you'll often find many individual objects all of the same kind.
- There may be thousands of other bicycles in existence, all of the same make and model.
- Each bicycle was built from the same set of blueprints and therefore contains the same components.
- In object-oriented terms, we say that your bicycle is an *instance* of the *class of objects* known as bicycles

Parameters

Parameters You've already seen examples of parameters, both in the Bicycle class and in the main method of the "Hello World!" application.

Recall that the signature for the main method is

public static void main(String[] args).

Here, the args variable is the parameter to this method. The important thing to remember is that parameters are always classified as "variables" not "fields".

Example

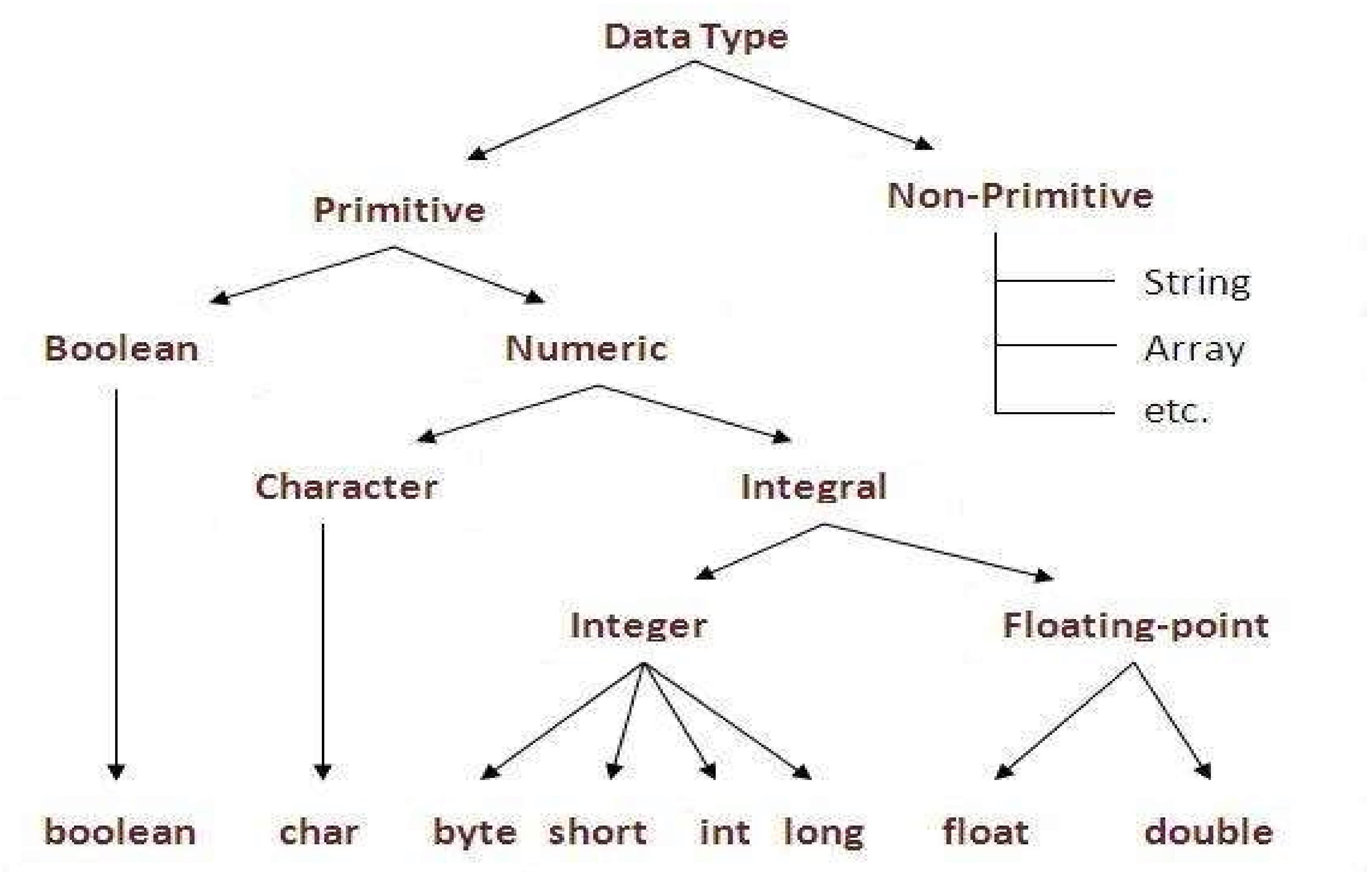
```
class A{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```

Naming

Every programming language has its own set of rules and conventions for the kinds of names that you're allowed to use:

1. Variable names are case-sensitive. A variable's name can be any legal identifier-an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "$", or the underscore character "_".
2. always begin your variable names with a letter, not "$" or "_". White space is not permitted.
3. Subsequent characters may be letters, digits, dollar signs, or underscore characters.
4. If the name you choose consists of only one word, spell that word in all lowercase letters. If it consists of more than one word, capitalize the first letter of each subsequent word. The names gearRatio and currentGear are prime examples of this convention

DATA TYPE


Data Type
Primitive
Non-Primitive
String
Array
etc.
Boolean
Numeric
Character
Integral
Integer
Floating-point
boolean
char
byte
short
int
long
float
double

Primitive Data Type

- The Java programming language is statically-typed, which means that all variables must first be declared before they can be used.
- The eight primitive data types supported by the Java programming language are:
- **byte**: The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive). The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters. They can also be used in place of int where their limits help to clarify your code; the fact that a variable's range is limited can serve as a form of documentation.

Primitive Data Type

- **short**: The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive). As with byte, the same guidelines apply: you can use a short to save memory in large arrays, in situations where the memory savings actually matters.
- **int**: By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} and a maximum value of $2^{31}-1$. In Java SE 8 and later, you can use the int data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of $2^{32}-1$. Use the Integer class to use int data type as an unsigned integer.

Primitive Data Type

- **long**: The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -2^{63} and a maximum value of 2^{63}-1. In Java SE 8 and later, you can use the long data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of 2^{64}-1. Use this data type when you need a range of values wider than those provided by int. The Long class also contains methods like compareUnsigned, divideUnsigned etc to support arithmetic operations for unsigned long.
- **boolean**: The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.

Primitive Data Type

- **float**: The float data type is a single-precision 32-bit IEEE 754 floating point. As with the recommendations for byte and short, use a float (instead of double) if you need to save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency. For that, you will need to use the java.math.BigDecimal class instead. Numbers and Strings covers BigDecimal and other useful classes provided by the Java platform.
- **double**: The double data type is a double-precision 64-bit IEEE 754 floating point. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.

Literal

- A *literal* is the source code representation of a fixed value; literals are represented directly in your code without requiring computation.
- As shown below, it's possible to assign a literal to a variable of a primitive type:

```
boolean result = true;
char capitalC = 'C';
byte b = 100;
short s = 10000;
int i = 100000;
```

Table shows the built-in data types with the size in bytes & default value

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Thank you