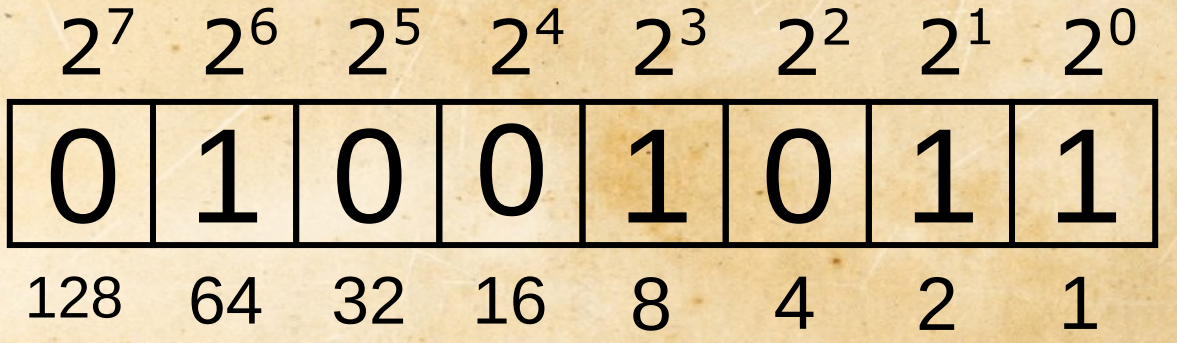
# Review

## Primitive Types and Variables

# Lecture Contents

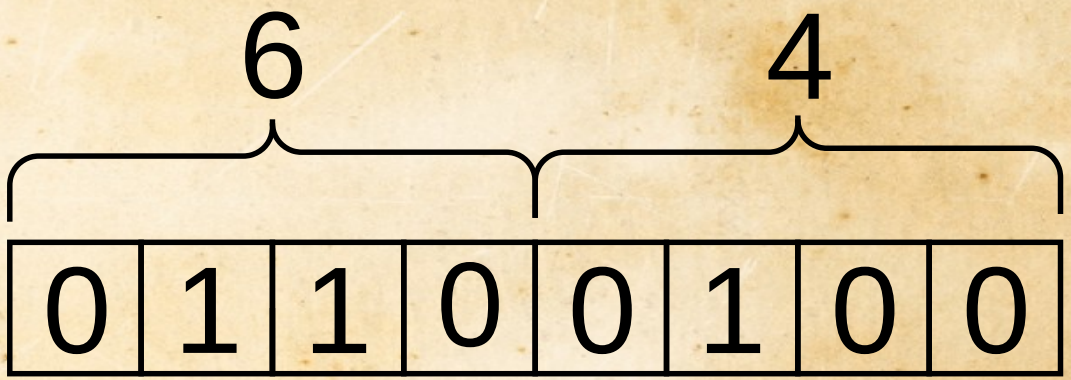
- Converting Numbers
  - Binary – Denary Conversion
  - Binary – Hexadecimal Conversion
- Java Types (AP Java Subset)
- Declaring and Initializing Variables
- Type Conversions
- Operations

# Binary – Denary Conversion




$$64 + 8 + 2 + 1 = 75$$

# Binary – Hexadecimal Conversion




| Decimal | Hexa-decimal | Binary |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

# Lecture Contents

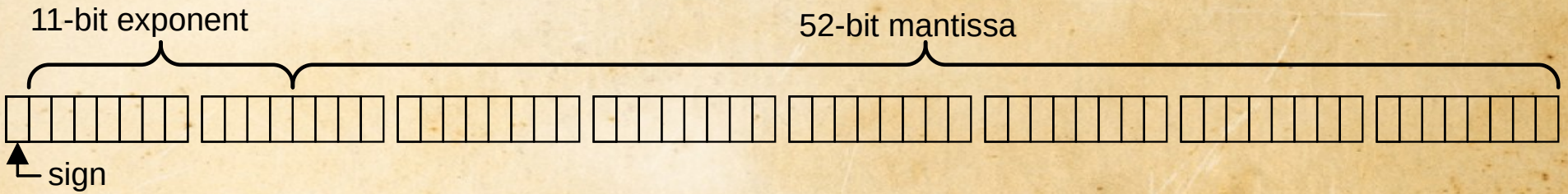
- Converting Numbers
  - Binary – Denary Conversion
  - Binary – Hexadecimal Conversion
- **Java Types (AP Java Subset)**
- Declaring and Initializing Variables
- Type Conversions
- Operations

# Java `int` type

- The Java type `int` is uses 32 bits and stores integer values.
- The Integer class has constants for the smallest and largest values:
  - `Integer.MAX_VALUE = +2,147,483,647`
  - `Integer.MIN_VALUE = –2,147,483,648`
-

# Java `double` type

- The Java type double uses 64 bits to store real numbers.
  - Format is similar to scientific notation.




$$\pm\ mantissa\ \times\ 2^{exponent}$$

# Java `char` type

- Java `char` type is ***not*** on the AP exam.
- Just know that Java characters are stored internally as numbers.
- Java `String` type is on the exam, but we will cover this later.

**ASCII Character Set** (0x20-0x7F)

| hex | char | hex | char | hex | char | hex | char | hex | char |
|---|---|---|---|---|---|---|---|---|---|
| 20 | *space* | 30 | 0 | 40 | @ | 50 | P | 60 | ` |
| 21 | ! | 31 | 1 | 41 | A | 51 | Q | 61 | a |
| 22 | “ | 32 | 2 | 42 | B | 52 | R | 62 | b |
| 23 | # | 33 | 3 | 43 | C | 53 | S | 63 | c |
| 24 | $ | 34 | 4 | 44 | D | 54 | T | 64 | d |
| 25 | % | 35 | 5 | 45 | E | 55 | U | 65 | e |
| 26 | & | 36 | 6 | 46 | F | 56 | V | 66 | f |
| 27 | ' | 37 | 7 | 47 | G | 57 | W | 67 | g |
| 28 | ( | 38 | 8 | 48 | H | 58 | X | 68 | h |
| 29 | ) | 39 | 9 | 49 | I | 59 | Y | 69 | i |
| 2A | * | 3A | : | 4A | J | 5A | Z | 6A | j |
| 2B | + | 3B | ; | 4B | K | 5B | [ | 6B | k |
| 2C | , | 3C | < | 4C | L | 5C | \ | 6C | l |
| 2D | - | 3D | = | 4D | M | 5D | ] | 6D | m |
| 2E | . | 3E | > | 4E | N | 5E | ^ | 6E | n |
| 2F | / | 3F | ? | 4F | O | 5F | _ | 6F | o |

# Java `boolean` type

- The Java `boolean` type can take only one of two values:
  - `true`
  - `false`
- Both `true` and `false` are keywords in Java; do not put them in quotation marks!

# List of Java Primitive Types

- *Integers*
  - `byte` (8 bits)
  - `short` (16 bits)
  - **`int`** (32 bits)
  - `long` (64 bits)
- *Real Numbers*
  - `float` (32 bits)
  - **`double`** (64 bits)
- *True/False*
  - **`boolean`** (1 bit?)
- *Letters*
  - `char` (16 bits)

**Note**: Primitive types tested in *AP Computer Science A* are given in bold font.

# Lecture Contents

- Converting Numbers
  - Binary – Denary Conversion
  - Binary – Hexadecimal Conversion
- Java Types (AP Java Subset)
- **Declaring and Initializing Variables**
- Type Conversions
- Operations

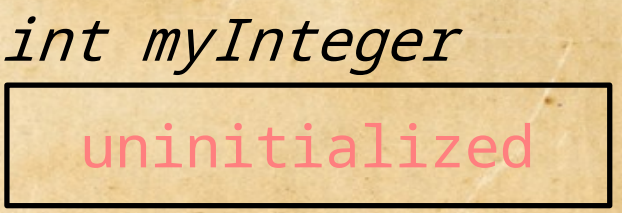
# Declaring a Variable

- Variables are ***declared*** with the syntax:

```
<type> <identifier>;
```

- Declaring a variable makes a place in memory to store a value.
- Examples:

```
int myInteger;
double myDouble;
boolean myBoolean;
```




- Remember that identifiers for variables use lower ***camel case***
  - the first letter of every word after the first is capitalized

# Initializing a Variable

- ***Initialize*** means ***assign a value*** for the first time.
- Variables can be *initialized* to a value using the ***assignment operator*** ( **=** ) with the following syntax:

```
<identifier> = <value>
```

- A variable must be declared before it is initialized:

***declaration***:

```
int myInteger;
```

*int myInteger*

uninitialized

***initialization***:

```
myInteger = -7;
```

*int myInteger*

-7

# Combining declaration and initialization

- ***Declaration*** and ***initialization*** can be combined into a single statement using the syntax:

```
<type> <identifier> = <value>;
```

- Examples:

```
int myInteger = 65535;
double pi = 3.14;
boolean isSad = false;
```

# Lecture Contents

- Converting Numbers
  - Binary – Denary Conversion
  - Binary – Hexadecimal Conversion
- Java Types (AP Java Subset)
- Declaring and Initializing Variables
- **Type Conversions**
- Operations

# Nuances of `int`

- Operations performed with integers remain integers.
  - For division, the result is ***truncated*** (cut off) at the decimal point

    In mathematics: `8 / 5 = 1.6`

    In Java: `8 / 5 = 1` ← ***truncated***, not rounded
- Recall that an `int` can only store integers, so with division, there may be a ***loss of precision*** in the answer.

# Operations with `double` and `int`

- With `double` and `int` as operands:
  - first, the `int` is automatically converted to a `double`
  - then the operation is performed
- Example:

| | *conversion* | | *operation* | |
|---|---|---|---|---|
| `8 / 5.0` | → | `8.0 / 5.0` | → | `1.6` |

- There is no loss of precision in this answer.

# `double` has limited precision

- The precision of `double` is 53 bits (binary digits)
- This is around 15 to 16 decimal digits.
- Example:

```
2.0 / 3.0 = 0.6666666666666666
```

← does not repeat infinitely

***Note***: there are other strange things that occurs due to loss of precision with different number systems, for example, in most computers, a rounding error occurs:

```
0.1 + 0.2 = 0.30000000000000004
```

This is beyond the scope of this course.

# Java `String` type

- The Java types `int`, `double`, and `boolean` are ***primitive types***.
- The Java `String` type is a more complex type called a ***reference type***.
- We will learn the difference later...

# Numbers and `String`

- Other mathematical operators cannot be used with strings, but the `+` operator performs ***concatenation*** (joining).

```
"Hello" + "World!"  →  "HelloWorld!"
```

- When a number and a `String` are operands to a + operator:
  - first, the number (`int` or `double`) is automatically converted to a `String`
  - then ***concatenation*** is performed
  - the result is of type `String`
- Example:
  - `"The answer is: " + 3.57  →  "The answer is 3.57"`
  - `String  double →  String`

# Numbers and `String`

- Multiple + operations are performed *left to right*

```
"The answer is: " + 3 + 5
                  ↓
"The answer is: " + "3" + 5
                  ↓
"The answer is: 3" + 5
                  ↓
"The answer is: 3" + "5"
                  ↓
"The answer is: 35"
```

# Numbers and `String`

- Order of Operations: `*` operator (multiplication) before `+`
  - `"The answer is: " + 3 * 5`

    ↓

    `"The answer is: " + 15`

    ↓

    `"The answer is: " + "15"`

    ↓

    `"The answer is: 15"`

- Only the `+` operator is valid for `String`, no other operator
  ( `*`, `/`, `-`, `%`, `>`, `<`, … )

# Lecture Contents

- Converting Numbers
  - Binary – Denary Conversion
  - Binary – Hexadecimal Conversion
- Java Types (AP Java Subset)
- Declaring and Initializing Variables
- Type Conversions
- **Operators**

# Operator Types

- ***Arithmetic***
- ***Assignment***
- *Comparison*
- *Logical*
- *Bitwise* ← not part of Java AP Subset

# Java Arithmetic Operators

- We should know the word "***operator***" from mathematics
- Java ***arithmetic operators***:
  - Add: `x + y`
  - Subtract: `x - y`
  - Multiply: `x * y`
  - Divide: `x / y`
  - Modulus: `x % y`

# Java Assignment Operators

- The operator = is the ***assignment operator***
  - The value on the right is written into the variable on the left

```
myInteger = -7;
```

# Java Assignment Operators

- Other ***assignment operators*** that combine ***arithmetic*** operations:

| | Arithmetic Operation and Assignment | Combined Operation |
|---|---|---|
| Addition | `x = x + 5` | `x += 5` |
| Subtraction | `x = x - 7` | `x -= 7` |
| Multiplication | `x = x * 3` | `x *= 3` |
| Division | `x = x / 6` | `x /= 6` |
| Modulus | `x = x % 4` | `x %= 4` |

- Note: There are also ***bitwise assignment*** operators, but those are not part of the AP Java Subset..

# Review

## Primitive Types and Variables