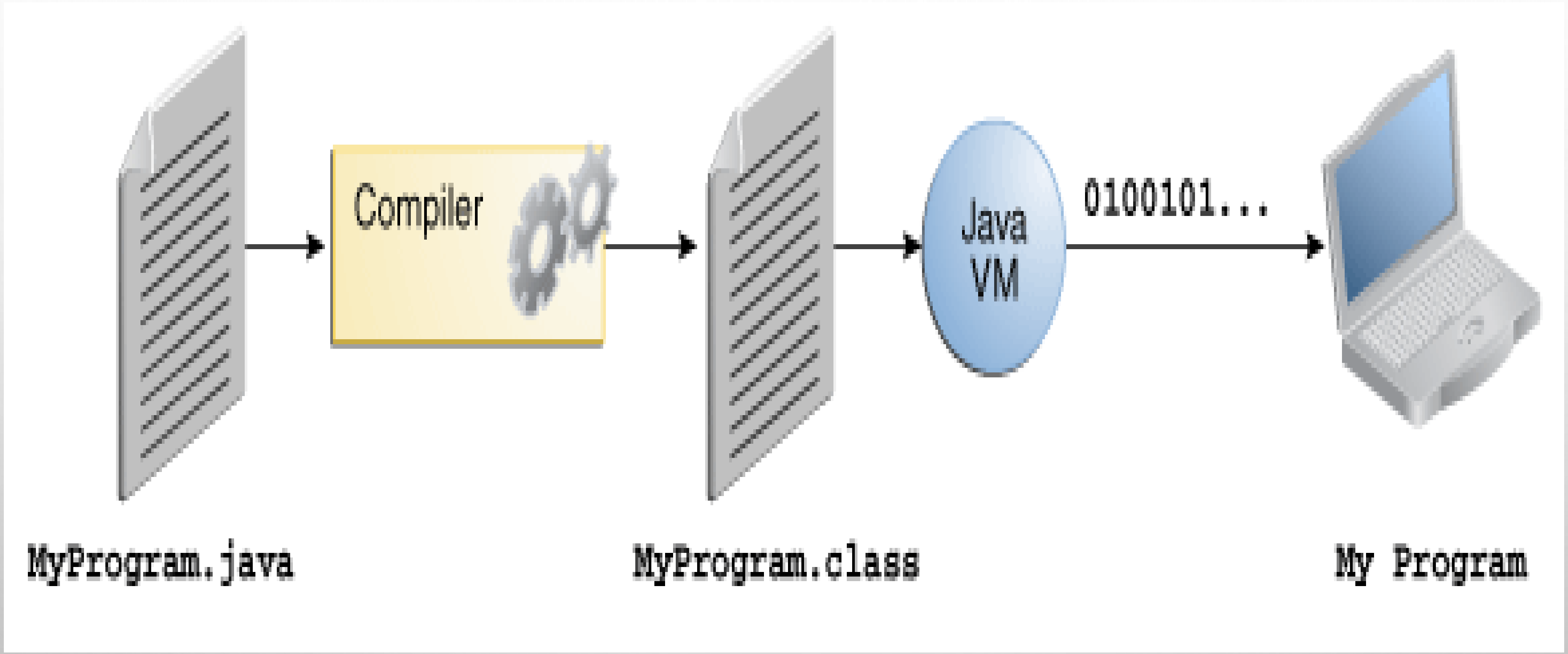
JAVA PROGRAMMING

JAVA TECHNOLOGY IS BOTH A PROGRAMMING LANGUAGE AND A PLATFORM.

- **THE JAVA PROGRAMMING LANGUAGE** IS A HIGH-LEVEL LANGUAGE THAT CAN BE CHARACTERIZED BY ALL OF THE FOLLOWING BUZZWORDS: (LINK)
  1. SIMPLE
  2. OBJECT ORIENTED
  3. DISTRIBUTED
  4. MULTITHREADED
  5. DYNAMIC
  6. ARCHITECTURE NEUTRAL
  7. PORTABLE
  8. HIGH PERFORMANCE
  9. ROBUST
  10. SECURE

- IN THE JAVA PROGRAMMING LANGUAGE, ALL SOURCE CODE IS FIRST WRITTEN IN PLAIN TEXT FILES ENDING WITH THE .JAVA EXTENSION.
- THOSE SOURCE FILES ARE THEN COMPILED INTO .CLASS FILES BY THE JAVAC COMPILER.
- A .CLASS FILE DOES NOT CONTAIN CODE THAT IS NATIVE TO YOUR PROCESSOR; IT INSTEAD CONTAINS BYTECODES — THE MACHINE LANGUAGE OF THE JAVA VIRTUAL MACHINE1 (JAVA VM).
- THE JAVA LAUNCHER TOOL THEN RUNS YOUR APPLICATION WITH AN INSTANCE OF THE JAVA VIRTUAL MACHINE.


Compiler
Java VM
0100101...
MyProgram.java
MyProgram.class
My Program

- BECAUSE THE JAVA VM IS AVAILABLE ON MANY DIFFERENT OPERATING SYSTEMS, THE SAME .CLASS FILES ARE CAPABLE OF RUNNING ON MICROSOFT WINDOWS, THE SOLARIS™ OPERATING SYSTEM (SOLARIS OS), LINUX, OR MAC OS.
- SOME VIRTUAL MACHINES, SUCH AS THE JAVA SE HOTSPOT AT A GLANCE, PERFORM ADDITIONAL STEPS AT RUNTIME TO GIVE YOUR APPLICATION A PERFORMANCE BOOST.
- THIS INCLUDES VARIOUS TASKS SUCH AS FINDING PERFORMANCE BOTTLENECKS AND RECOMPILING (TO NATIVE CODE) FREQUENTLY USED SECTIONS OF CODE.

- **THE JAVA PLATFORM:** A *PLATFORM* IS THE HARDWARE OR SOFTWARE ENVIRONMENT IN WHICH A PROGRAM RUNS.
- MOST PLATFORMS CAN BE DESCRIBED AS A COMBINATION OF THE OPERATING SYSTEM AND UNDERLYING HARDWARE.
- THE JAVA PLATFORM DIFFERS FROM MOST OTHER PLATFORMS IN THAT IT'S A SOFTWARE-ONLY PLATFORM THAT RUNS ON TOP OF OTHER HARDWARE-BASED PLATFORMS.
- THE JAVA PLATFORM HAS TWO COMPONENTS:
    1. THE *JAVA VIRTUAL MACHINE*
    2. THE *JAVA APPLICATION PROGRAMMING INTERFACE* (API)
- THE API IS A LARGE COLLECTION OF READY-MADE SOFTWARE COMPONENTS THAT PROVIDE MANY USEFUL CAPABILITIES.
- IT IS GROUPED INTO LIBRARIES OF RELATED CLASSES AND INTERFACES; THESE LIBRARIES ARE KNOWN AS *PACKAGES*.

- AS A PLATFORM-INDEPENDENT ENVIRONMENT, THE JAVA PLATFORM CAN BE A BIT SLOWER THAN NATIVE CODE. HOWEVER, ADVANCES IN COMPILER AND VIRTUAL MACHINE TECHNOLOGIES ARE BRINGING PERFORMANCE CLOSE TO THAT OF NATIVE CODE WITHOUT THREATENING PORTABILITY.
- THE TERMS "JAVA VIRTUAL MACHINE" AND "JVM" MEAN A VIRTUAL MACHINE FOR THE JAVA PLATFORM.

WHAT CAN JAVA TECHNOLOGY DO

- THE GENERAL-PURPOSE, HIGH-LEVEL JAVA PROGRAMMING LANGUAGE IS A POWERFUL SOFTWARE PLATFORM. EVERY FULL IMPLEMENTATION OF THE JAVA PLATFORM GIVES YOU THE FOLLOWING FEATURES:
  - **DEVELOPMENT TOOLS**: THE DEVELOPMENT TOOLS PROVIDE EVERYTHING YOU'LL NEED FOR COMPILING, RUNNING, MONITORING, DEBUGGING, AND DOCUMENTING YOUR APPLICATIONS. AS A NEW DEVELOPER, THE MAIN TOOLS YOU'LL BE USING ARE THE JAVAC COMPILER, THE JAVA LAUNCHER, AND THE JAVADOC DOCUMENTATION TOOL.

- **APPLICATION PROGRAMMING INTERFACE (API)**: THE API PROVIDES THE CORE FUNCTIONALITY OF THE JAVA PROGRAMMING LANGUAGE. IT OFFERS A WIDE ARRAY OF USEFUL CLASSES READY FOR USE IN YOUR OWN APPLICATIONS. IT SPANS EVERYTHING FROM BASIC OBJECTS, TO NETWORKING AND SECURITY, TO XML GENERATION AND DATABASE ACCESS, AND MORE. THE CORE API IS VERY LARGE; TO GET AN OVERVIEW OF WHAT IT CONTAINS, CONSULT THE JAVA PLATFORM STANDARD EDITION 8 DOCUMENTATION.
- **DEPLOYMENT TECHNOLOGIES**: THE JDK SOFTWARE PROVIDES STANDARD MECHANISMS SUCH AS THE JAVA WEB START SOFTWARE AND JAVA PLUG-IN SOFTWARE FOR DEPLOYING YOUR APPLICATIONS TO END USERS.
- **USER INTERFACE TOOLKITS**: THE JAVAFX, SWING, AND JAVA 2D TOOLKITS MAKE IT POSSIBLE TO CREATE SOPHISTICATED GRAPHICAL USER INTERFACES (GUIS).
- **INTEGRATION LIBRARIES**: INTEGRATION LIBRARIES SUCH AS THE JAVA IDL API, JDBC API, JAVA NAMING AND DIRECTORY INTERFACE (JNDI) API, JAVA RMI, AND JAVA REMOTE METHOD INVOCATION OVER INTERNET INTER-ORB PROTOCOL TECHNOLOGY (JAVA RMI-IIOP TECHNOLOGY) ENABLE DATABASE ACCESS AND MANIPULATION OF REMOTE OBJECTS.

“HELLO WORLD” APP (LINK)

- COMPILING: **JAVAC** HELLOWORLDAPP.JAVA
- RUNNING: **JAVA -CP** . HELLOWORLDAPP

LANGUAGE BASICS

- OBJECTS:
  - SOFTWARE OBJECTS ARE CONCEPTUALLY SIMILAR TO REAL-WORLD OBJECTS: THEY TOO CONSIST OF **STATE** AND RELATED **BEHAVIOR**.
  - AN OBJECT STORES ITS STATE IN *FIELDS* (VARIABLES IN SOME PROGRAMMING LANGUAGES) AND EXPOSES ITS BEHAVIOR THROUGH *METHODS* (FUNCTIONS IN SOME PROGRAMMING LANGUAGES).
  - METHODS OPERATE ON AN OBJECT'S INTERNAL STATE AND SERVE AS THE PRIMARY MECHANISM FOR OBJECT-TO-OBJECT COMMUNICATION.
  - HIDING INTERNAL STATE AND REQUIRING ALL INTERACTION TO BE PERFORMED THROUGH AN OBJECT'S METHODS IS KNOWN AS ***DATA ENCAPSULATION*** — A FUNDAMENTAL PRINCIPLE OF OBJECT-ORIENTED PROGRAMMING.

VARIABLES

- THE JAVA PROGRAMMING LANGUAGE DEFINES THE FOLLOWING KINDS OF 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); THE CURRENT SPEED OF ONE BICYCLE IS INDEPENDENT FROM THE CURRENT SPEED OF ANOTHER.
  - **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** 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.
- **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". THIS APPLIES TO OTHER PARAMETER-ACCEPTING CONSTRUCTS AS WELL (SUCH AS CONSTRUCTORS AND EXCEPTION HANDLERS) THAT YOU'LL LEARN ABOUT LATER IN THE TUTORIAL.

VARIABLE NAMING

- **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 "$", OR THE UNDERSCORE CHARACTER "_".
  - THE CONVENTION, HOWEVER, IS TO ALWAYS BEGIN YOUR VARIABLE NAMES WITH A LETTER, NOT "$" OR "_".
  - ADDITIONALLY, THE DOLLAR SIGN CHARACTER, BY CONVENTION, IS NEVER USED AT ALL. YOU MAY FIND SOME SITUATIONS WHERE AUTO-GENERATED NAMES WILL CONTAIN THE DOLLAR SIGN, BUT YOUR VARIABLE NAMES SHOULD ALWAYS AVOID USING IT.
  - A SIMILAR CONVENTION EXISTS FOR THE UNDERSCORE CHARACTER; WHILE IT'S TECHNICALLY LEGAL TO BEGIN YOUR VARIABLE'S NAME WITH "_", THIS PRACTICE IS DISCOURAGED.
  - **WHITE SPACE IS NOT PERMITTED**.

- **SUBSEQUENT CHARACTERS**
  - MAY BE LETTERS, DIGITS, DOLLAR SIGNS, OR UNDERSCORE CHARACTERS.
  - CONVENTIONS (AND COMMON SENSE) APPLY TO THIS RULE AS WELL.
  - WHEN CHOOSING A NAME FOR YOUR VARIABLES, USE FULL WORDS INSTEAD OF CRYPTIC ABBREVIATIONS.
  - DOING SO WILL MAKE YOUR CODE EASIER TO READ AND UNDERSTAND.
  - IN MANY CASES IT WILL ALSO MAKE YOUR CODE SELF-DOCUMENTING; FIELDS NAMED CADENCE, SPEED, AND GEAR, FOR EXAMPLE, ARE MUCH MORE INTUITIVE THAN ABBREVIATED VERSIONS, SUCH AS S, C, AND G.
  - ALSO KEEP IN MIND THAT THE NAME YOU CHOOSE MUST **NOT** BE A **KEYWORD** OR **RESERVED** WORD.

- 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.
- IF YOUR VARIABLE STORES A CONSTANT VALUE, SUCH AS STATIC FINAL INT NUM_GEARS = 6, THE CONVENTION CHANGES SLIGHTLY, CAPITALIZING EVERY LETTER AND SEPARATING SUBSEQUENT WORDS WITH THE UNDERSCORE CHARACTER.
- BY CONVENTION, THE UNDERSCORE CHARACTER IS NEVER USED ELSEWHERE.

JAVA LANGUAGE KEYWORDS (LINK)

- YOU CANNOT USE ANY OF THE FOLLOWING AS IDENTIFIERS IN YOUR PROGRAMS.
- THE KEYWORDS CONST AND GOTO ARE RESERVED, EVEN THOUGH THEY ARE NOT CURRENTLY USED.
- TRUE, FALSE, AND NULL MIGHT SEEM LIKE KEYWORDS, BUT THEY ARE ACTUALLY LITERALS; YOU CANNOT USE THEM AS IDENTIFIERS IN YOUR PROGRAMS.

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

PRIMITIVE DATA TYPES (LINK)

- THE JAVA PROGRAMMING LANGUAGE IS STATICALLY-TYPED, WHICH MEANS THAT ALL VARIABLES MUST FIRST BE DECLARED BEFORE THEY CAN BE USED.
- THIS INVOLVES STATING THE VARIABLE'S TYPE AND NAME:

```
INT GEAR = 1;
```

- DOING SO TELLS YOUR PROGRAM THAT A FIELD NAMED "GEAR" EXISTS, HOLDS NUMERICAL DATA, AND HAS AN INITIAL VALUE OF "1".

- A VARIABLE'S DATA TYPE DETERMINES THE VALUES IT MAY CONTAIN, PLUS THE OPERATIONS THAT MAY BE PERFORMED ON IT.
- IN ADDITION TO INT, THE JAVA PROGRAMMING LANGUAGE SUPPORTS **SEVEN** OTHER PRIMITIVE DATA TYPES.
- A PRIMITIVE TYPE IS PREDEFINED BY THE LANGUAGE AND IS NAMED BY A RESERVED KEYWORD.
- PRIMITIVE VALUES DO NOT SHARE STATE WITH OTHER PRIMITIVE VALUES.
- 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.

- **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.
  - STATIC METHODS LIKE COMPAREUNSIGNED, DIVIDEUNSIGNED ETC HAVE BEEN ADDED TO THE INTEGER CLASS TO SUPPORT THE ARITHMETIC OPERATIONS FOR UNSIGNED INTEGERS.

- **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.
- **FLOAT**: THE FLOAT DATA TYPE IS A SINGLE-PRECISION 32-BIT IEEE 754 FLOATING POINT.
  - ITS RANGE OF VALUES IS BEYOND THE SCOPE OF THIS DISCUSSION, BUT IS SPECIFIED IN THE FLOATING-POINT TYPES, FORMATS, AND VALUES SECTION OF THE JAVA LANGUAGE SPECIFICATION. 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.
  - ITS RANGE OF VALUES IS BEYOND THE SCOPE OF THIS DISCUSSION, BUT IS SPECIFIED IN THE FLOATING-POINT TYPES, FORMATS, AND VALUES SECTION OF THE JAVA LANGUAGE SPECIFICATION.
  - 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.
- **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.
- **CHAR**: THE CHAR DATA TYPE IS A SINGLE 16-BIT UNICODE CHARACTER.
  - IT HAS A MINIMUM VALUE OF '\U0000' (OR 0) AND A MAXIMUM VALUE OF '\UFFFF' (OR 65,535 INCLUSIVE).