CS102A Introduction to Computer Programming

Course Review

Disclaimer

- Course content that does not appear in the slides may still be tested in the final exam

Computers and Programs

Computer Organization

- Input unit
- Output unit
- Memory unit (内存，主存)
- Central processing unit (CPU, 中央处理器)
  - Arithmetic and logic unit (ALU, 算术逻辑单元)
  - Control Unit (CU, 控制单元)
- Secondary storage unit (辅助存储单元，二级存储器)

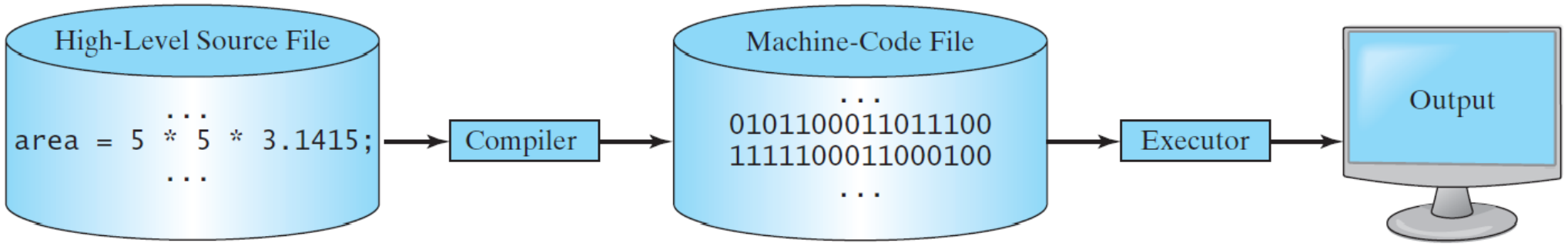
John von Neumann
(1903-1957)
Hungarian-American
mathematician, physicist

Computer Program

- A computer program is a set of machine-readable instructions that tells a computer how to preform a specific task
- Programs are written in programming languages
- There are many programming languages
  - Low-level, understandable by a computer
  - High-level, needs a translator (compiler)!

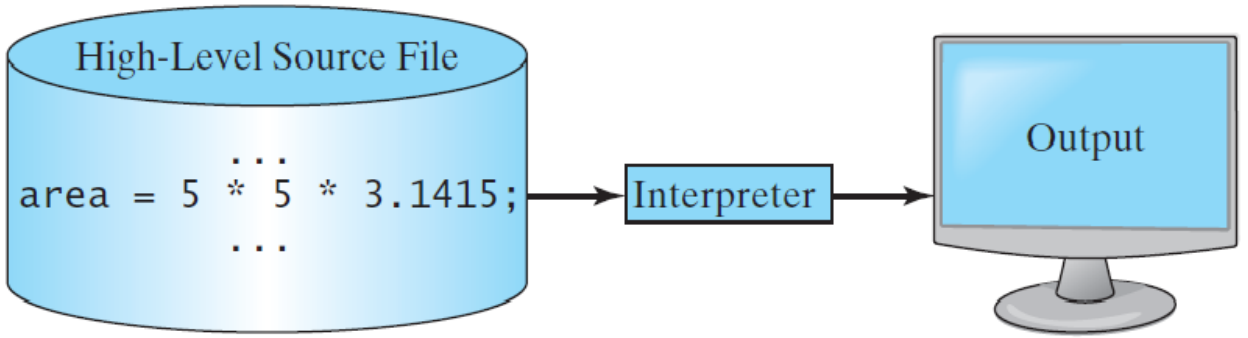
Compilation: from source to executables

- A complier translates source programs into machine codes that can run directly on the target computer.



Interpreter

- An interpreter directly translates and executes the statements in source code, without requiring the programs to be compiled into machine codes.



Computer Software

A set of programs, also including libraries and non-executable data, e.g., documentation

Application software: Programs designed to perform specific tasks and typically very easy to use

- MS Word, PowerPoint, Chrome, Photoshop, WeChat etc.

System software: Programs that support the execution/development of other programs.

- Operating systems (e.g., Windows, Mac OSX, Linux, iOS, Android)
- Translation systems (e.g., compilers, linkers, assemblers)

Java Programs

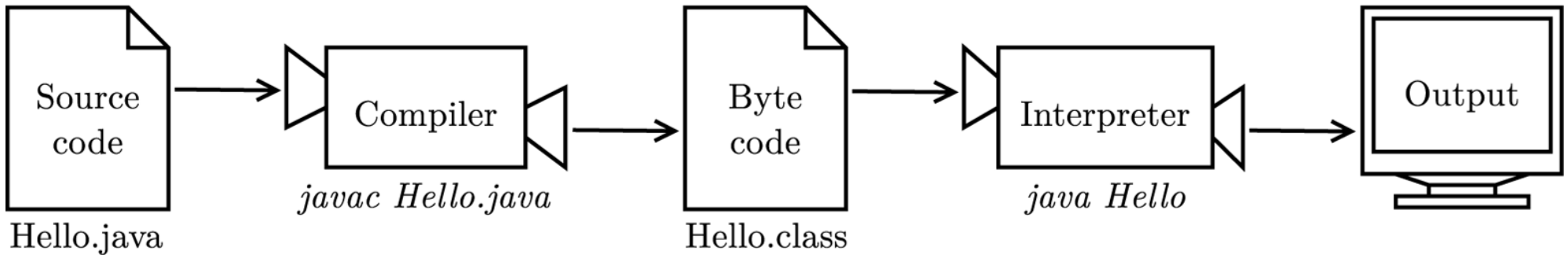
Java

- Java is a general-purpose computer-programming language
- Java was originally developed by **James Gosling** at Sun Microsystems (acquired by Oracle) and released in 1995

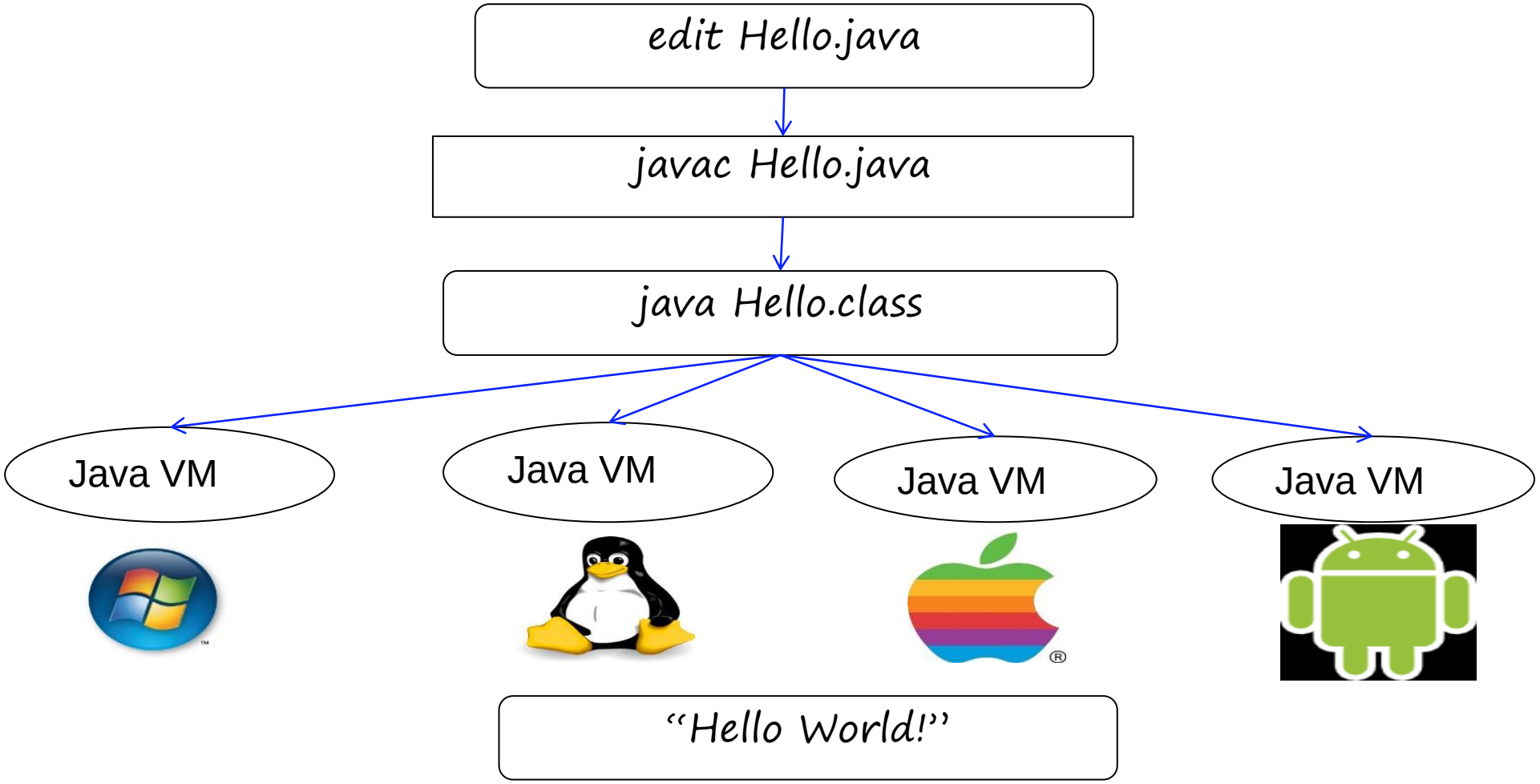
JDK, JRE, and JVM

- **Java Development Kit (JDK)** is a software development environment for developing Java programs. It includes a Java Runtime Environment (JRE), an interpreter (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools.
- **Java Runtime Environment (JRE)** provides the minimum requirements for executing a Java program; it consists of a Java Virtual Machine (JVM), core classes, and supporting files.
- **Java Virtual Machine (JVM)** is an abstract computing machine that enables a computer to run a Java program.
- JDK = JRE + Development tools, JRE = JVM + Library classes

Java is both compiled and interpreted



Java is platform-independent



Java Language Basics

Identifiers

- A name in a Java program is called an **identifier**, which is used for identification purpose. It can be a class name, a method name, a variable name, or a label name
- The only allowed characters in Java identifiers are **a to z, A to Z, 0 to 9, $** and _(Underscore).
- Identifiers can't start with digit, e.g., **123name** is not valid.
- Java identifiers are case sensitive, e.g.,, **name, Name, NAME** are different

Identifiers cont.

- We can't use reserved keywords as identifiers, i.e., int if = 20 will not compile

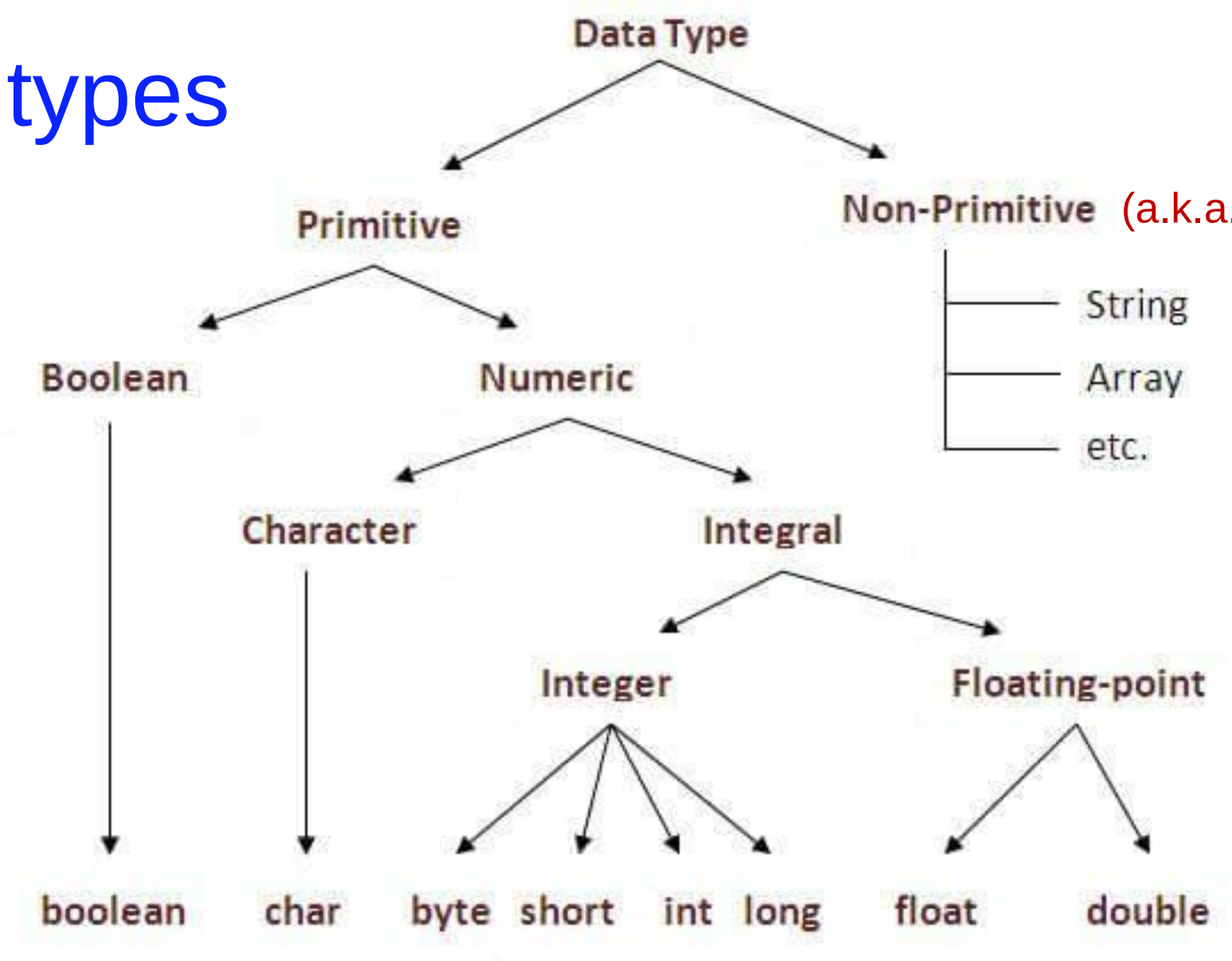
abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile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

Keyword names are all in lower case

Variable

- A variable is a container which holds data while a Java program executes
- It is the name of the memory location allocated for storing the data
- There are three kinds of variables in Java
  - Local variables: A variable declared inside a method (be careful with the scope)
  - Instance variable: A variable declared inside a class but outside the methods and not declared static
  - Static variable: A variable which is declared as static

Data types



https://www.javatpoint.com/variable-datatype

More on reference types

- All non-primitive types are reference types, including instantiable classes and arrays
  - `Scanner, String, String[], int[]`
- Programs use reference-type variables to store the locations of objects in memory.
  - `GradeBook myGradeBook = new GradeBook(“CS102A”);`
  - Such a variable is said to refer to an object in the program.
- Reference-type variables, if not explicitly initialized, get the default value `null`

Common arithmetic operators

Operator	Description
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1

- Integer division yields an integer quotient. The fractional part is simply discarded (3 / 2 = 1)
- int a = 6; int b = ++a; int c = a++; (a will be 8, b and c will be 7 after execution)

Equality and relational operators

Operator	Description
==	Equal to (do not confuse with the assignment operator =)
!=	Not equal to
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal

Common logical operators

- **Logical operators** help form complex conditions by combining simple ones:
  - && (conditional AND, **short-circuit behavior**)
  - || (conditional OR, **short-circuit behavior**)
  - & (boolean logical AND)
  - | (boolean logical inclusive OR)
  - ^ (boolean logical exclusive OR)
  - ! (logical NOT)

Structured Programming

- Only three forms of control are needed to implement any algorithm:
  - Sequence
  - Selection
  - Repetition

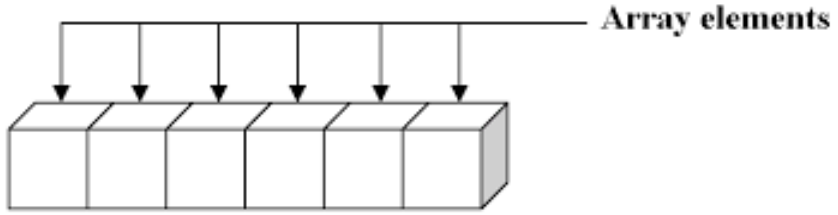
Structured Programming

- Selection is implemented in one of three ways:
  - `if` statement (single selection)
  - `if…else` statement (double selection)
  - `switch` statement (multiple selection)
- The simple `if` statement is sufficient to provide any form of selection. Everything that can be done with the `if…else` and `switch` can be implemented by combining `if` statements.

Structured Programming

- Repetition is implemented in one of three ways:
  - `while` statement
  - `do…while` statement
  - `for` statement
- The `while` statement is sufficient to provide any form of repetition. Everything that can be done with `do…while` and `for` can be done with the `while` statement.
- Differences between `break` and `continue` statements.

Arrays



- An **array** is a group of variables (elements) containing values of the same type.
- Arrays are objects, so they're considered reference types. Arrays are created by the keyword `new`
- Elements can be either primitive types or reference types. They can be retrieved using indexes
- Valid array indexes: 0 to array.length - 1 (runtime exception will occur when using other values)
- Unlike collections such as ArrayList, the capacity of an array is fixed once created

Strings

- A string is a sequence of characters
- A string is an object of the class `java.lang.String`
- Strings can be created by using string literals or various String constructors
  - `String s = “hello world”; String s1= new String(“hello world”);`
- String objects are immutable. Any modification creates a new String object.
- String class provides many useful methods: length(), charAt(), substring()
- The equals() method tests whether two strings are identical (not ==, which only works for primitive types)

Enum types

- Enum is a special data type that enables a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.
- All enum types are reference types
- Enum constants are implicitly `final` and `static`
- Any attempt to create an object of an enum type with operator `new` results in a compilation error (enum constructor should be private or package-private)
- Besides enum constants, enums can also contain members such as constructors, fields and methods

Classes and Objects

Classes and Objects

- **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

- **Object** − An object is an instance of a class. Objects have states and behaviors.
  - Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating.

Classes

- A class can contain member variables (instance and static ones) and methods (instance and static ones)

- Differences between instance variables/methods and static variables/methods

Access Level Modifiers

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Note that this is for controlling access to class members. At the top level, a class can only be declared as `public` or package-private (no explicit modifier)

Methods

- Method parameters are local variables (only visible in the method)
- **Parameter vs. arguments:** A parameter is the variable which is part of the method's declaration. An argument is an expression used when calling the method.
- When calling methods, all arguments are **passed by value**. A method call can pass two types of values to the called method: copies of primitive values and copies of references to objects.

Method overloading

- Methods of the same name can be declared in the same class, as long as they have different sets of parameters
- Used to create several methods that perform the same/similar tasks on different types or different numbers of arguments
- Compiler distinguishes overloaded methods by their **signature**: a combination of the method's name and the number, types and order of its parameters (return type is not part of method signature)

Constructors

- A constructor is a special method used to initialize the data of an object
- Java compiler will create a default no-argument constructor for a class that has no explicitly-defined constructors
- The compiler will not create a default no-argument constructor for a class that has any explicitly-defined constructor
- Constructors can be overloaded to allow different ways of object creation

The `super` keyword

- The main use of the `super` keyword is to invoke a superclass constructor in a subclass constructor

- "`super`" can also be used to invoke overridden methods of the superclass (when the subclass overrides a method of its superclass)

The `this` keyword

- The main use of `this` reference is to differentiate the formal parameters of methods and the fields of classes (when the fields are shadowed by the method parameters)

- "`this`" can also be used to invoke another constructor of the same class in the body of one constructor

Inheritance and polymorphism

Inheritance

- Java only allows single inheritance, but one class can implement several interfaces

```
public class SubClass extends SuperClass implements Interface1, Interface2 { … }
```

- A subclass inherits all of the public and protected members of its parent, no matter what package the subclass is in
- If the subclass is in the same package as its parent, it also inherits the package-private members of the parent
- Private members of the superclass cannot be inherited, constructors are not class members and cannot be inherited
- Every class directly or indirectly extends `java.lang.Object`

Method Overriding

- A subclass can override a method it inherits from its parent to provide more specific implementation (final methods cannot be overridden, private methods and static methods are implicitly final)
- An overridden method must have the same signature as the superclass method
- *An overridden method may have a more specific return type
- *The access level of an overriding method can be higher, but not lower than that of the overridden method (package-private < `protected` < `public`)

Polymorphism

- In Java, polymorphism is the ability of an object to take different forms. All java objects are polymorphic and can pass multiple IS-A tests

- With polymorphism, an object of a subclass can be treated as an object of the superclass

- Objects of different types can be accessed through the same interface. Each type can provide its own implementation of this interface

Assignment between superclass and subclass variables

- Assigning a superclass object's reference to a superclass variable is natural
- Assigning subclass object's reference to a subclass variable is natural
- Assigning a subclass object's reference to a superclass variable is safe (the superclass variable can be used to access only superclass members)
- Assigning a superclass object's reference to a subclass variable leads to compilation errors

Dynamic binding (or late binding)

- When Java compiler encounters a method call made through a reference variable, it determines if the method can be called by checking the variable's class type
  - If that class contains the proper method declaration (or inherits one), the call will be successfully compiled
- At execution time, the type of the object to which the variable refers determines the actual method to use (dynamic binding)

Static binding (or early binding)

- In Java, final methods in a super class cannot be overridden in the subclass. Private methods and static methods are implicitly final.

- A final method's declaration cannot change so the calls to the final methods are resolved statically at compile time (static binding)

Abstract class

- A class that represents a generalization and provides functionality but is only intended to be extended and not instantiated
- An abstract class can contain instance/static variables/methods and constructors
- Usually contain one or more abstract methods that are intended to be overridden by subclasses
- Subclass can become concrete only if it implements all inherited abstract methods; otherwise, it has to be declared as abstract

Abstract class

- Although abstract classes cannot be used to instantiate objects, they can be used to declare variables. Abstract superclass variables can hold references to objects of any concrete classes derived from them.

- Note that we can use abstract class names to invoke static methods declared in those abstract classes (since invoking static methods do not require the existence of objects)

Interface

- An interface describes a set of methods that can be called on an object, but does not provide concrete implementations for all the methods

- An interface is often used when unrelated classes need to share common methods and constants (interfaces do not enforce a class relationship)

- An interface is a reference type. When a class implements an interface, it has an IS-A relationship with the interface data type

Interface declaration

- An interface contains only constants and abstract methods (Java 7 or earlier*)
  - All fields are implicitly public static final
  - All methods are implicitly public abstract
- An interface can extend multiple interfaces
- An interface cannot have a constructor
- An interface can be used to define variables. In such cases, any object you assign to the variable must be an instance of a class that implements the interface

* Default methods and static methods in interfaces are allowed since Java 8

Exception Handling

Exception

- An exception is an indication of a problem that occurs during a program's execution. It would disrupt the normal flow of instructions
- In Java, only `Throwable objects` can be used with the exception handling mechanism
- `Try-catch-finally` statement and how program control flows

Checked exceptions vs. unchecked exceptions

- All exception types that are direct or indirect subclasses of the class `RuntimeException` are **unchecked exceptions**, which are often caused by programmer's mistakes (defects in the code).

- All exception types that inherit from the class `Exception` but not `RuntimeException` are **checked exceptions**, which are often caused by conditions that are not under control of the program
  - Compiler enforces a catch-or-declare policy for checked exceptions

Generic class/method and collections

Generics

- Generic class and methods in Java parameterize data types that can be used in the methods or class
- They help write general code that can handle any data types or implement data structures in a type independent manner
- They help make your program more type safe (compilers does type inference and inserts safe cast operations when necessary)