Terms	
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
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
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 $2^{31}-1$.
long	The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -263 and a maximum value of 263-1
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
double	64 bit floating point
boolean	true or fals
char	single 16 bit unicode character
Super Class	Parent Class (Each Child can only have one parent SuperClass)
Extends	Inherit from parent class
fields	member variables in a class
local variables	variables in a method or block of code
parameters	variables in method declarations
public	accessible from all classes
private	only accessible from within class
Overloading methods	methods can have same name if they take different parameters
Overriding	In sub class has same method, name, parameters and return type but you change the behavior of the method in the child class
	You cannt override a private or static method in Java
type	all variables must have a type
this	reference to current object
declaring a constant	static final double PI = 3.14
Enum	A collection of Constants
= =	equality operator
Set	Cannot contain duplicate values
Map	Associative Array

Modifier	**Class**	**Package**	**Subclass**	**World**
public	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	No
no modifier	Yes	Yes	No	No
private	Yes	No	No	No

Lambda Function

Lambda Function	You want to pass functionality as an argument to another function
	If your anonymous class needs to be simpler you can use a lambda function
	Small anonymous functions or functions without a name
	throw away functions
	You are passing a single unit if behavior you want to pass to other code
	One issue with anonymous classes is that if the implementation of your anonymous class is very simple, such as an interface that contains only one method, then the syntax of anonymous classes may seem unwieldy and unclear. In these cases, you're usually trying to pass functionality as an argument to another method, such as what action should be taken when someone clicks a button. Lambda expressions enable you to do this, to treat functionality as method argument, or code as data.

Abstract

Abstract Class	A class that is declared abstract
	It may or may not include abstract methods
	They can not be instantiated
	Similar to interfaces,
	May have static fields or static methods
Abstract Method	Method that is declared without an implementation
	If the class includes abstract method it must be declared abstract
*an interface is more of an empty shell, like a contract. An abstract class is an actual class	

Static

Static Method	Belongs to the class, it can be called without instantiating the class
Static Variable	Belongs to class calling will always put this value out

Interfaces

Interfaces	Only contains the method signature (no body) and fields (variables)
	So when you make one it says all the classes need to have a certain method with that method built in the class that implements it
	Can only contain constants, method signatures, default methods, static methods and nested types
	They can be implemented by classes or extended by other interfaces
	method signatures have no braces and are terminated with a semicolon.
Example	int changeLanes(Direction direction, double startSpeed, double endSpeed);

Classes: Inheritance

Inheritance	Inherit common state and behavior from parent class
	Public class MountainBike extends Bicycle {}
Object	The topmost java class
Class Vs Interface	One significant difference between classes and interfaces is that classes can have fields whereas interfaces cannot
	Can only extend one class but can have multiple interfaces
Overriding	An instance method in a subclass with the same signature (name, plus the number and the type of its parameters) and return type as an instance method in the superclass overrides the superclass's method
Interface Methods	Default methods and abstract methods in interfaces are inherited like instance methods. However, when the supertypes of a class or interface provide multiple default methods with the same signature, the Java compiler follows inheritance rules to resolve the name conflict. These rules are driven by the following two principles
Nested Class	A class within a class can be static or non-static
	Non-static nested classes (inner class) has acces to other members of enclosing class even if they are declared private
	Static nested classes do not have access to other members of the enclosing class
Inner Class	Associated with instance of enclosing class so it has direct access to that objects methods and fields
Anonymous Class	Anonymous classes enable you to make your code more concise. They enable you to declare and instantiate a class at the same time. They are like local classes except that they do not have a name. Use them if you need to use a local class only once.
	class declared within the body of a method without naming the class
	Help make code more concise
Local Class	class declared within the body of a method

Class: Exception

Exception	Used to handle errors and exceptional events
Three Types	checked, error, runtime
Exception	Occurs during execution of program and disrupts the normal flow of the programs instructions
Overview	When error occurs within method the method creates an object and hands it off to the runtime system
	This object is called and exception object and contains information about the error
	This is called throwing an exception
	After the method throws an exception the runtime system attempts to find something to handle it
	It searches the call stack for a method that can handle the exception (exception handler)
Try Catch	Code that might throw certain exceptions must either be enclosed with try catch or throws

```
try {
   } catch (ExceptionType name) {
   } catch (ExceptionType name) {
} finally {}
```

finally	finally always executes when the try block exits

Handling Errors

Three Types	checked, error, runtime
checked exception	A well written program should anticipate and recover from these
	Asking for a file and expecting things can go wrong
	All exceptions are checked exceptions except those indicated by *Error* and *RuntimeExcpetion* and their subclasses
error	Exceptional conditions external to application that it cannot anticipate or recover from
	Opens a program but there is a system or hardware error
	These are not subject to catch or specify requirement
runtime exception	Exceptional conditions internal to the program
	Could be logic error, programming bugs or improper use of an API

Memory

Stack Memory	store local variables and function calls
	Part of memory where temporary variables are stored
	Used to execute a threa
	When a method is invoked it creates a new blcok in that stack
Heap Memory	Store objects and can become full when it is garbage collection is initiated
Thread	Each thread can have its own stack

Concurrency

Concurrency	A program often does many things at the same time
Compute System	Normally has many processes and threads running
Process	Has a self contained excecution environment
	Has its own memory space
	Usually synoymous with programs or applications
	What appears to be a single application could be cooperating processes
	Most instances of JVM run as single process
Thread	Sometimes called a lightweigh process
	Both processes and threads provide an execution environment, but creating a new thread requires fewer resources than creating a new process.
	Threads exist within a process — every process has at least one. Threads share the process's resources, including memory and open files. This makes for efficient, but potentially problematic, communication.
	Multithreaded execution is an essential feature of the Java platform. Every application has at least one thread — or several, if you count "system" threads that do things like memory management and signal handling. But from the application programmer's point of view, you start with just one thread, called the main thread.
Thread.sleep	Pause current execution
Interrupt	An interrupt is an indication to a thread that it should stop what it is doing and do something else

Collection

Collection	A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.
	The Collection interface (java.util.Collection) and Map interface (java.util.Map) are the two main “root” interfaces of Java collection classes.

Terms

Big O	Worst Case Scenario

O(1)	
O(1)	Executes in the same time regardless of the size of the input data

O(N)	
O(N)	Performance will grow linearly and in direct proportion to the input data set
	Searching for a matching string in a for loop
	Could be found early but this always assumes the upper limit

O(N^2)	
O(N^2)	$O(N^2)$ represents an algorithm whose performance is directly proportional to the square of the size of the input data set. This is common with algorithms that involve nested iterations over the data set. Deeper nested iterations will result in $O(N^3)$, $O(N^4)$ etc.

O(2^N)	
O(2^N)	O(2N) denotes an algorithm whose growth doubles with each additon to the input data set. The growth curve of an O(2N) function is exponential - starting off very shallow, then rising meteorically. An example of an O(2N) function is the recursive calculation of Fibonacci numbers:

Searching	
Linear	if you have 100 items to search then the worst case scenario would require that you look at every item in the input before you came across your desired value. It is called linear because the time is takes to search is exactly correlated with the amount of items in the search
	simple no complex algorithms
Binary Search	Split into two and look for value, split again (like a phonebook)