What is Java? What are the benefits of Java?
High-level, strongly typed, compiled OOP Language
platform independent; garbage collection done for you

What is the difference between the JRE, JDK, and JVM?
Java Runtime Environment: contains all runtime libraries that code will be calling and using; needed to run the code
Java Development Kit: developer tools you need to write/compile the code like a compiler, debugger, documentation tools (javadoc), and other command-line utilities.
Java Virtual Machine: interpreter; compiles bytecode into native language for the operating system.

JRE contains JVM.
JDK contains JRE

What happens during the compilation process?
Compilation results in translating code into bytecode.

For example, in Java, that happens when .java compiles into .class. The compiler translates the .class into bytecode.

TEXTBOOK: “Java source code is compiled into bytecode when we use the javac compiler. The bytecode gets saved on the disk with the file extension .class. When the program is to be run, the bytecode is converted, using the just-in-time (JIT) compiler. The result is machine code which is then fed to the memory and is executed.”

What is a constructor?

In Java, a constructor is a block of codes similar to the method that has no output type. It is called when an instance of the object is created, and memory is allocated for the object. If a programmer does not explicitly create a constructor, then the compiler injects a “default, no-args” constructor.

Types:

1. **Default**: automatically added by the compiler if no constructor provided

```
public Example () {


}
```

2. **Parameterized constructor**: has arguments or values. Multiple values can be passed in. Can be called using initializer list. Used when a programmer wants to initialize a class with default values.

```
public Example (String a, String b, String c) {
        Some class.methods();
}
```

3. **No-args**: constructor with no parameters

```
public Example () {

        Some class.methods();
        Any instance variables to instantiate

}
```

What are the primitive data types?
Each variable specifies its own type.
There are two categories of variables: primitives and reference variables.

Primitives
A primitive variable uses a predefined amount of space in memory
There are 8 primitive datatypes:

1. ***boolean***: 1-bit true or false
2. ***byte***: 8-bit two's complement integer
3. ***short***: 16-bit two's complement integer
4. ***char***: 16-bit Unicode character
5. ***int***: 32-bit integer
6. ***float***: 32-bit floating-point number
7. ***long***: 64-bit integer
8. ***double***: 64-bit floating-point number

Reference Variables
A reference variable uses a class name as its type.
Ex: Dog d = new Dog();

Arrays
An array is a data structure that acts as a collection of either primitive or a reference-type variable. In Java, the size of an array is fixed upon creation.

TEXTBOOK: An array is a contiguous block of memory storing a group of sequentially stored elements of the same type.

What is a no args constructor?

No-args constructor with no parameters

```
public Example () {

        Some class.methods();
        Any instance variables to instantiate

}
```

What is the default constructor?

The default constructor is a no-args constructor that the Java compiler inserts on your behalf; it contains a default call to super(); which is the default behavior. If you implement any constructor then you no longer receive a default constructor.

What are the scopes of a variable in java?

1) Instance, or object, scope
2) Method scope
3) Block scope
4) Class, or static, scope

Block scope example:

```
for (int i = 0; i < 234234; i++) {

}
```

i is only seen inside the for loop block

Instance:

```
public class  Ex () {

        private String s; <- each object instantiated will have its own s String which can be accessed by any of its methods

        public Ex (String s) {
                this.s = s;
        }
}
```

Method:

```
Public void methodName() {
        String a = “sdfsdf”;
}
```

String a is only visible inside methodName

What are the different access modifiers?

- There are two types of modifiers in java: **access modifiers** and **non-access modifiers**.
- The access modifiers in java specify accessibility (scope) of a data member, method, constructor or class.

There are 4 types of java access modifiers:

- private
- default (also referred to as package private)
- protected
- public

There are many non-access modifiers such as static, abstract, synchronized, native, (OBSOLETE: volatile & transient) etc. Here, we will learn access modifiers.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

What are the different control statements and how are they different?

Java programming language provides the following types of loop to handle looping requirements:

- If-else statement
- While loop: if the condition is false it will skip the loop
    - while(Boolean_expression) {

      // Statements to run if boolean is true
    - }

- For loop
    - for(int i=0;i<a.length;i++) //length is the property of array
    - System.out.println(a[i]);
- Do..While loop: runs the condition at least once before checking the while condition
- Switch statement
- Branching statements
    - break
    - continue
    - return

How do you create an array in java?

int a[]=new int[5]; //declaration and instantiation - creates an integer array of size 5

a[0]=10;

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

State the type of array you want (string, int, etc.) and then put square brackets after it.

Int[] b = {0,1,2,3,4,5}

What are varargs?

Variable arguments are used in place of the square brackets in the main method so the programmer can set an argument whose size is determined at runtime. Only one vararg parameter can be used in a method, and they MUST be the last parameter defined.

<u>Example:</u>
Usual: public static void main(String[]args) {}

Varargs: public static void main(String… args) {}

public int add (int a, int… b) {

}

class.add(3, 4, 5, 6); // a = 3, b = [4, 5, 6]

What are packages and imports?

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two forms: built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql, etc.

What is a static import?
import static java.lang.Math.*;
Allows you to access its static members without having to call it as (example) Math.sqrt(4);
-> just sqrt(4);

Feature that allows direct access to static members of a class without needing to state the class name or object. Example: Instead of using Math.floor(), we can declare floor() directly.

What is static?

If you declare any variable as static, it is known as a static variable and it's declared as a member of a certain class.

- The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
- The static variable gets memory only once in the class area at the time of class loading.

What are Strings?

Strings, which are widely used in Java programming, are an immutable sequence of characters. In the Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings.

What are some string methods?
https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

.charAt();
.length();
.concat();

.equals(); // every object has a .equals();
.indexOf();
.toUpperCase();
.toLowerCase();
.toCharArray();
.subString();
.trim();

What is the difference between String, StringBuilder, and StringBuffer?

1. String is immutable/unchangeable - this means every time you want to change a string, you create an entirely new string because you cannot change the contents of a string
2. StringBuilder is not thread-safe, but fastest
3. StringBuffer is thread-safe, slower than StringBuilder

What is the String Pool?

Special location in the memory heap for strings (because strings take up so much memory). Unless you use the ***new*** keyword to create a new String, then a reference pointer will point to the same string in memory if the strings match. String pools do not allow duplicates without forcing instantiation using the ***new*** keyword.

What is the difference between the stack and the heap?

Stack stores method invocations and reference variables in stack frames. Multiple stacks are possible and every time a thread is created, it's given its own stack.

Heap is a space in memory where all your variables are; keeps track of variables and objects.

What is an exception?

Exceptions are events that occur during the execution of programs that disrupt the normal flow of instructions (e.g. divide by zero, array access out of bound, etc.).

In Java, an exception is an **object** that wraps an error event that occurred within a method and contains:

- Information about the error including its type
- The state of the program when the error occurred
- Optionally, other custom information

What is the difference between exception and error?

Errors are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

An error is an unsolvable problem while the exception is a solvable problem.

What are the ways one can handle an exception?

Use a try/catch/finally block to handle an exception.

```
try {
        //code that may throw exception
} catch (Exception e){
         //code to execute if exception is caught
} finally {
        //code to execute at the end if an exception is thrown or not
}
```

A try block is used to contain code that can potentially throw an exception.

A catch block is used to catch any exceptions that may be thrown from code in the try block. Afterwards, it will execute any code within its block.

A finally block is used to execute code regardless of whether an exception is thrown (even if a return statement is in the try block). Finally blocks do not execute if the program calls System.exit().

You can also **throw** an exception by handing the exception back to the code that called it (this might cause the program to terminate, however).

NOTE: You can also catch RuntimeException or a subclass if you wanted.

What are checked exceptions and unchecked exceptions?

Exceptions come in two forms: checked and unchecked.

1. Checked exceptions are tested by the compiler. You need to handle these with code or you'll not be able to compile your program.

2. Unchecked exceptions are not tested by the compiler. They happen during runtime.

How many catch blocks can be used in a try catch?
You can use as many catch blocks for as many conditions as you can think of for exceptions. However, catch blocks must go from more specific exceptions to more general exceptions.

What does the finally block do?
A finally block is used to execute code regardless of whether an exception is thrown (even if a return statement is in the try block). Finally blocks do not execute if the program calls System.exit().

How do I create a custom exception?
Create a new class by extending the exception class, can be used in a try/catch/throw block

Example:

```
public class IncorrectFileNameException extends Exception {
   public IncorrectFileNameException(String errorMessage) {
      super(errorMessage);
   }
}
```

What is autoboxing?
A Java feature that automatically changes primitives to a wrapper class implicitly

Map<int, int> // not valid (int is a primitive)
Map<Integer, Integer> // (Integer is a wrapper class for an int)

https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html

What is a wrapper class?
A class which allows programmers to treat/convert primitives as objects. Used for certain methods which only accept objects but not primitives.

What is garbage collection?
Frees up memory used by an object that does not have a reference variable “pointing” to it

What are the pillars of object oriented programming?

Object Oriented Programming Pillars

- Inheritance
- Polymorphism
- Encapsulation - bundling everything together into a “capsule”
- Abstraction - abstracting away/hiding implementation details

Inheritance

- **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).
- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.
- Inheritance represents the **IS-A relationship** which is also known as a parent-child relationship.

Advantages of Inheritance

- Method Overriding (so runtime polymorphism can be achieved).
- Code Reusability.

Types Of Inheritance In Java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only.

Polymorphism

- Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.
- Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.

Encapsulation

- **Encapsulation in Java** is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.
- We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

- The **Java Bean** class is the example of a fully encapsulated class

Abstraction

- ☐ The “abstracting” away of details and data
- ☐ The “blackbox” method: having input(s) into a system that gives output(s); the means to which the system achieves its goals are generally unknown/don’t matter

What is the difference between an abstract class and an interface?

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class, it is implemented by a class.
- An interface can extend multiple interfaces.

What are the differences between FileinputStream, FileReader, and bufferedReader (and their output counterparts)?

What are generics? Why use them?

What is the difference between Comparator and Comparable?

What is the purpose of the Object class?

What is the difference between == and .equals?

What is the purpose of hashcode?

What is JUnit?

What are the annotations of JUnit?

What are the different assert methods of JUnit?

How do I create a test case and test suite in JUnit?

What is Maven?

What is the maven lifecycle?

What is the purpose of the pom.xml?

What are the purposes of a maven repository among projects?