1. **What is the difference between JDK, JRE, and JVM?**

JVM

JVM is an acronym for Java Virtual Machine; it is an abstract machine which provides the runtime environment in which Java bytecode can be executed. It is a specification which specifies the working of Java Virtual Machine. Its implementation has been provided by Oracle and other companies. Its implementation is known as JRE.

JVMs are available for many hardware and software platforms (so JVM is platform dependent). It is a runtime instance which is created when we run the Java class. There are three notions of **the** JVM: specification, implementation, and instance.

JRE

JRE stands for Java Runtime Environment. It is the implementation of JVM. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

JDK

JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

* Standard Edition Java Platform
* Enterprise Edition Java Platform
* Micro Edition Java Platform

JDK, JRE and JVM?

• Java Virtual Machine: JVM is an abstract machine. It actually runs by Java code. Most people know Java with this slogan "Write once and run everywhere' This slogan is because of JVM.

• Java Runtime Environment: JRE is what we need to run a Java program and contains set of libraries and other files that JVM uses at run time. JRE = JVM + Library Classes

• Java Development Kit: JDK is what we need to compile Java source code and contains JRE, development tools. JDK = JRE + Development tools

1. **What are the various access specifiers (aka access modifiers) in Java?**

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

* **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
* **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
* **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.
* **Private** The private class, methods, or variables defined as private can be accessed within the class only.

### What is the purpose of static methods and variables?

The methods or variables defined as static are shared among all the objects of the class. The static members belong to the class itself, not to any single object. As a result, we do not need to create an object to access those static variables or methods. It can be done directly through the class name. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class or perform any action that does not rely on an object (i.e. utility methods)

For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as **static**.

### Can we override the static methods?

No, we cannot override static methods. If a static method is created with a similar method signature as a method being passed down by inheritance this will be method hiding, not overriding.

### What is the static block?

The static block is a code block that allows only static members. This block is always executed once and before anything else. This block is run whenever that specific class is loaded for the first time. This block is commonly used to initialize static variables.

### What is the difference between static (class) method and instance method?

|  |  |
| --- | --- |
| Static | Instance |
| Uses keyword static | No extra keywords |
| Can be called by the class name so an object does not need to be created to use it | Must make an object to use the method |
| Can be used in other static context: methods, blocks, etc.. | All static and non-static fields can be used in the instance methods. |
| Ex: public static int cube (int n) {return n\* n \* n;} | Ex: public void msg() {…} |

### Why is the main method static?

Because an object is not required to call static methods. If we make the main method non-static, the JVM will have to create an object first and then call main() method which will lead to the extra memory allocation. The main method being static also allows us to call other static variable or methods.

### Can we execute a program without main() method?

Yes, one of the ways to execute the program without the main method is using static block. Also, in testing we use other approaches such as the Test annotations to run our code.

### What is an object?

The Object is the real-time entity having some state and behavior. In Java, an Object is an instance of the class which has the instance variables as the properties and the methods as the behaviors. The object of a class can be created by using the **new** keyword.

* What if the static modifier is removed from the signature of the main method?

Program compiles. However, at runtime, it throws an error "NoSuchMethodError."

### Object vs Class

Class is a blueprint or template which you can create as many objects as you like.

* + The blueprint will define the information (instance variables) or actions (methods) that an object would have

Object is an instance of a class

Class is declared using class key word, Object is created through new keyword mainly.

### What is the difference between stack and heap?

Both are used for memory but have different purposes

Stack: Stores primitives, method calls, and references

Heap: Stores the objects themselves. The String pool is also in the heap

### Can we overload the constructors?

Yes, the constructors can be overloaded by changing the number of parameters or by changing the data type of the parameters.

### Can we make constructors static?

As we know that the static members (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

### What is the method?

A definition of actions that are executed whenever called. Methods allow these actions/statement to be created once and called any number of times. This makes the code reusable.

### Difference between a Constructor and a Method?

* Constructor does not have a return type and constructor’s name must be same as the class name.
* Constructor is called automatically when a new object is created. Constructor is invoked implicitly.
* The Java compiler provides a default constructor if we do not have any constructor.
* Constructors are not inherited by child classes
* Method have a return and the method’s name may or not be same as the class name o Methods is invoked explicitly.
* Method is not provided by compiler in any case.
* Methods are inherited by child classes.

### What is immutable?

* After object creation, the object cannot be altered/changed. If the object needs to be changed a new object must be made.

### What is an Array?

* + An array is a data structure that holds a fixed number of values of a single type (primate and objects). The length of an array is established when the array is created. After creation, its length is fixed.
  + Each item in an array is called an element, and each element is accessed by its numerical index. The index numbers begin with 0. The last index of any given array would always be the length - 1
  + **Advantage of Java Array** 
    - Code Optimization: It makes the code optimized, we can retrieve or sort the data easily.
    - Random access: We can get any data located at any index position.
* **Disadvantage of Java Array**
* Size Limit: We can store only fixed size of elements in the array. It does not grow its size at runtime. To solve this problem, collection framework is used in java.

### Array vs ArrayList

Array is a part of core Java programming and ArrayList is part of collection framework.

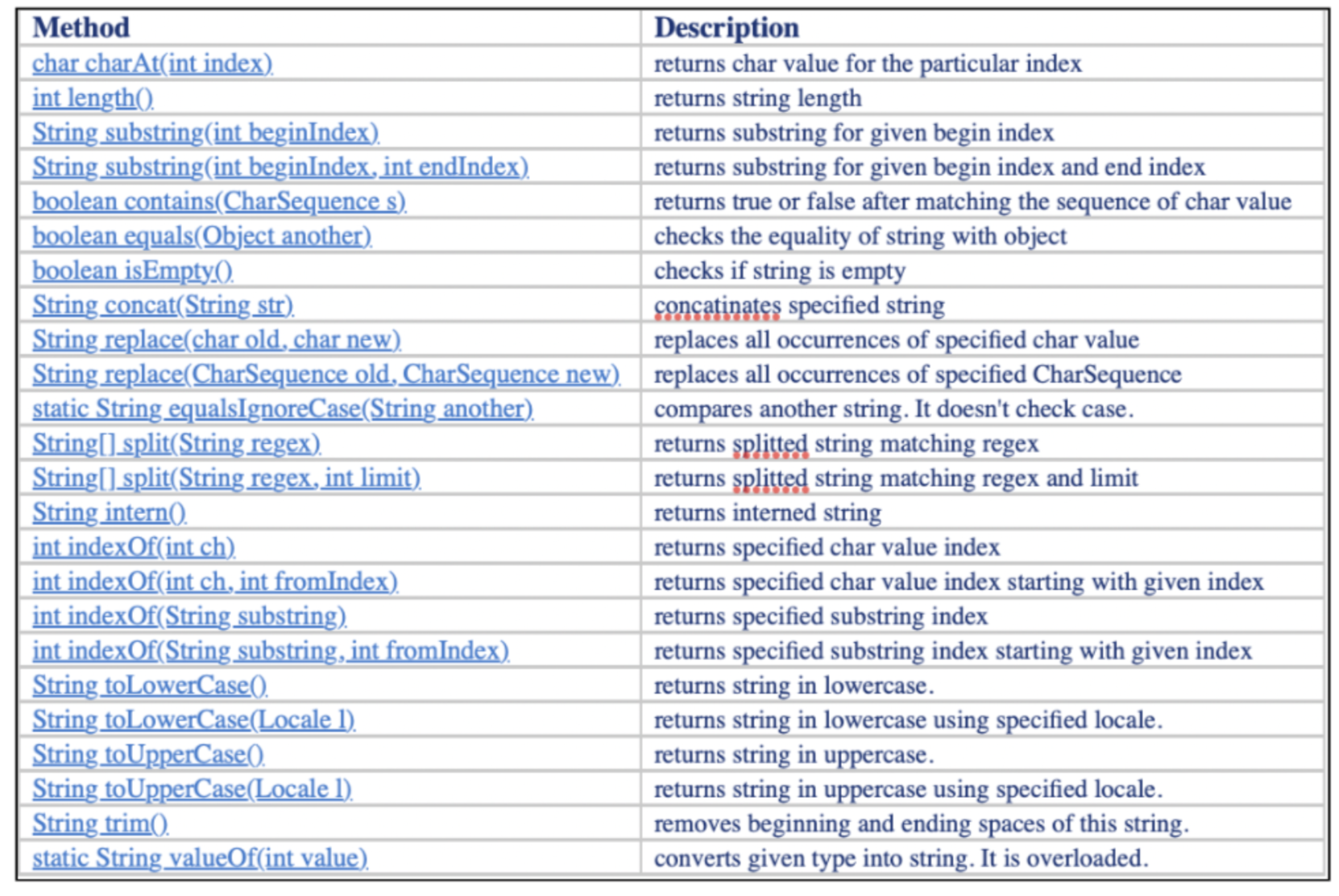
• Major difference is that Array is a fixed length data structure, so we can’t change length, but ArrayList is re-sizeable.

• The other major one is that Array can contain both primitive and object elements, but ArrayList can only contain only objects. It cannot contain primitive types.

### What are Wrapper classes?

Wrapper classes are object representations of primitive datatypes. These classes are needed to be able to store primitive like data in collections such as ArrayList. These wrapper classes also have useful utility methods.

### Important String Methods?

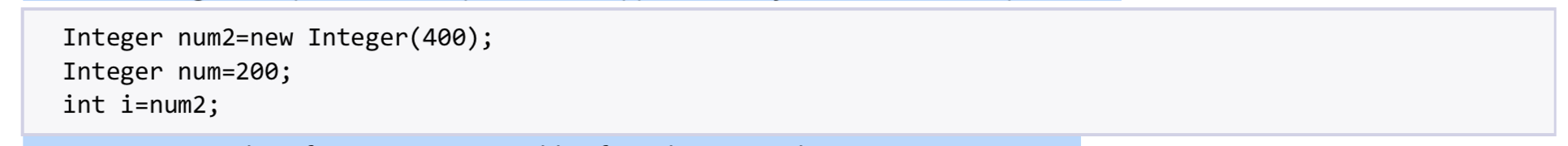


### Do you know typecasting? What is casting?

* + Auto-boxing is a process that will convert(cast) a primate datatype to an object wrapper class object automatically.



* + Un-boxing is a process when the wrapper class object is converted(cast) to a private datatype automatically.

Assigning a value of one type to a variable of another type is known as Type Casting.

### What are exceptions in java and how do you handle them?

Exceptions occur when there is an issue with the program. There are two types of exceptions: Checked exceptions (aka Compile time) and Unchecked exceptions (aka Runtime). All checked exceptions must be handled otherwise the program will not compile. Unchecked exceptions happen during runtime so if they occur the program will be interrupted during execution making these exceptions are more difficult to handle. Exceptions can be handled using the try/catch block which lets you define the actions taken if specific exceptions occur. It is also possible to use the keyword throws in the method signature to allow the program to compile and accept that an exception may occur, but this is not handling the exception.

1. **Difference between Error and Exception in Java?**
   1. Both Error and Exception are derived from Throwable in Java.
   2. Error represent errors which generally cannot be handled. They are issues outside of the code
      1. For example: OutOfMemoryError, NoClassDefFoundError
   3. Exception represent problems coming from the code which can be handled.
      1. For example: IOException, NullPointerException
   4. Exceptions are divided in two categories checked and unchecked Exceptions. Checked Exceptions must be handled before execution, but Unchecked Exception occur during runtime
   5. Errors are something developer are not required to do anything with
   6. In general Errors are beyond anyone’s control, but Exceptions can be guessed and handled
2. **Difference between throw and throws in Java?**
   1. throw and throws are two keywords related to Java Exceptions
   2. throw keyword is used to throw an exception explicitly and throws keyword is used to allow an exception to be thrown if it occurs
   3. throw keyword is used in a code block while throws is used in method signature to allow the exception.
   4. throw keyword can also be used to break a switch statement without using break keyword

### Difference between RuntimeException and CheckedException in Java?

* 1. Checked Exceptions must be handled before execution, but Unchecked Exception occur during runtime

Common exceptions: NullPointerExceptio, ArrayIndexOutOfBound, ClassNotFoundException, IOException.

1. **What is the collection framework in java?**

The collection framework is a group of interfaces and concrete classes that are different data structures used to solve different problem.

 **java.util.Collection** - interface which defines the basis actions of a collection

 **Set** *(Unique things)* - DOES NOT ALLOW DUPLICATES. Classes that Implement Set:

* **HashSet:** Used when you don't want any duplicates and you don't care about order when you iterate through
  + Unordered and Unsorted
* **LinkedHashSet:** Ordered version of HashSet and use over HashSet when you care about iteration order

### SortedSet

* **TreeSet**: Elements will be in ascending order, according to the natural order of the elements
  + Can also customize constructor to implement your own rules of the natural order

 **List** *(list of things)* - cares about the index. Classes that implement List:

* **ArrayList:** Fast iteration and fast random access and ordered (by index)
* **LinkedList:** Ordered by index position and elements are doubly-linked to one another
  + It is a good choice for implementing stack and queue
  + Iterates more slowly than ArrayList but fast insertion and deletion
* **Vector**: Same as ArrayList BUT vector methods are synchronized (thread-safe)

 **java.util.Collections** - a class that holds static utility methods for use with collections; Includes add, remove, contains, size, and iterator, etc.

1. **What are Maps?**

**Map** *(things with unique ID):*

Important: none of the Map-related classes and interfaces extend form Collection. The implementation classes of Map are thought of “collections”, not Collection. Classes that implement Map:

* **HashMap:**  Unsorted and unordered data structure that allows one null KEY and multiple null values in a collection. Works by entries, which are key/value pairs

### Hashtable

* + Same as HashMap BUT HashTable methods are synchronized (REMEMBER. ONLY METHODS ARE SYNCHRONIZED, NOT CLASSES OR VARIABLES)
  + Hashtable won't let you have anything NULL (NO NULLS AT ALL)

### LinkedHashMap

* + Maintains insertion order (or optionally, access order)
  + Slower than HashMap for adding/removing elements but FASTER ITERATION
* **SortedMap:**
  + TreeMap: Keys are sorting in natural order

**OOP Interview Questions**

1. **What is Object Oriented Programming (OOP)?**
   1. OOP is a programming language model organized around object rather than actions (logic and functions).
   2. In other words, OOP mainly focuses on the objects that are required to be manipulated instead of logic. This approach is ideal for the programs large and complex codes and needs to be actively updated or maintained.
   3. It makes development and maintenance easier - It provides data hiding - It provides ability to simulate real-world.

**OOP language follow 4 principles**:

* + 1. **Encapsulation:** We can hide direct access to data by using private keyword and we can access private data by using getter and setter method.
    2. **Abstraction:** It is a process of hiding implementation details and showing only functionality to the user. Abstraction lets you focus on what the object does instead of how it does it.
    3. **Inheritance:** It is used to define the relationship between two classes. Inheritance is when a child class acquires all properties and behaviors from a parent class. Child class can reuse all the codes written in parent class. It provides the code reusability.
    4. **Polymorphism:** It is the ability of object to behave in multiple form. The most common use of polymorphism in Java is when a parent class reference type of variable is used to refer to a child class object.

*Example*

WebDriver driver = new ChromeDriver();

We use method overloading and overriding to achieve Polymorphism.

### What is encapsulation and how did you use it?

* 1. Data hiding by making variables private and providing public getter and setter methods.
  2. In my project I created multiple POJO/BEAN classes to manage test data and actual data.
* Ex: I take JSON from API response and convert to object of my POJO class all variables are private with getters and setter.

### What is the concept of Abstraction?

* 1. In OOP, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.
  2. In other words, the user will have the information on what the object does instead of how it does it.
  3. In Java, abstraction is achieved using Abstract classes and interfaces.
  4. For example: when you log in to your bank account online, you enter your user-id and password and press the login. What happens then, how the input data sent to the server, how it gets verified are all abstracted away from you.

### Difference between Abstract Class and Interface?

* 1. Main difference is methods of a Java interface are implicitly abstract and cannot have implementations. A Java abstract class can have instance methods that have implementation.
  2. A class that is declared with abstract keyword, is known as abstract class. It can have abstract and non--abstract methods.
  3. An Interface is a blueprint of a class. It is a template, and it is declared with interface keyword. It can have abstract methods, default methods, static methods, and public final static variables
  4. When we want to use Abstract class, we use “extend” keyword. When we want to use Interface, we use “implement” keyword.
  5. Abstract class and interface both are used to achieve abstraction. Both cannot be instantiated 🡺 we cannot create an object from them.

### What is Inheritance?

* 1. Inheritance represents the **IS-A** relationship which is also known as a parent-child relationship.
  2. It is the mechanism in java by which one class can inherit the features (fields and methods) of another class.
  3. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.
  4. You can add new methods and fields in your current class also.
  5. Code reusability is the most important benefit of inheritance because subclasses inherits the variables and methods of superclass.

### What is Polymorphism?

* 1. Polymorphism is a very important concept in OOP because:
     1. it enables to change the behavior of the applications in the run time based on the object on which the invocation happens.
     2. an object can have different forms
  2. Two types: **Compile Time** which is Static and **Run Time** Polymorphism which is related with child and parent class.
  3. Polymorphism is implemented using the concept of Method overloading and method overriding. This can only happen when the classes are under the parent and child relationship using inheritance.

### Difference between method Overloading and method Overriding?

* 1. First and most important difference between overloading and overriding:
     1. in case of overloading, method name must be the same, but the parameters must be different.
     2. in case of overriding, method name and parameters must be same
     3. In method overloading, return type can be same or different. In method overriding, return type must be same or covariant type
  2. Second major difference between method overloading and overriding:
     1. We can overload method in the same class but method overriding occurs in two classes that have inheritance relationship.
  3. We cannot override static, final, or private method in Java, but we can overload static, final, and private method in Java.

1. **Difference between Object and Class?**
   1. Class is a blueprint or template which you can create as many objects as you like Object is a member or instance of a class
   2. Class is declared using class keyword, Object is created through new keyword mainly.

A class is a template for objects. A class defines object properties including a valid range of values, and a default value. A class also describes object behavior. An object is a member or an "instance" of a class and has states and behaviors in which all its properties have values that you either explicitly define or that are defined by default settings.

Class - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support. If we compare them there are many differences but let me tell you some of them which are important to know;

* 1. There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization. There is only one way to define class in java using class keyword.
  2. Object is created many times as per requirement. Class is declared once.
  3. Object is an instance of a class. Class is a blueprint or template from which objects are created.

1. **How do you use an abstract class in your project?** 
   1. These concepts are commonly used in framework development. Abstract class is used in defining a common super class while writing Page Object Model layer of the framework. We usually create an abstract class named BasePage to have all common members for every page written in this class example **getPageTitle().**
   2. Then each Page class (HomePage, LoginPage, DashboardPage etc.) inherit from BasePage. Sometimes one may need to change the behavior of methods implemented in superclass. So, subclass has freedom to override that method where we use polymorphism. This is how we use Abstract class in real projects.
2. What is the difference between HashMap and Hashtable?

HashMap is not synchronized; Hashtable is synchronized.

HashMap can contain one null key and multiple null values; Hashtable cannot contain any null key or null value.

HashMap is not thread-safe, so it is useful for non-threaded applications; Hashtable is thread-safe, and it can be shared between various threads.

1. What is the difference between Collection and Collections?

The Collection is an interface whereas Collections is a class.

The Collection interface provides the standard functionality of data structure to List, Set, and Queue. However, Collections class is to sort and synchronize the collection elements.

The Collection interface provides the methods that can be used for data structure whereas Collections class provides the static methods which can be used for various operation on a collection.

1. Important terminology for inheritance:
   1. **Class:** the group of objects which have common properties. It is a template or blueprint from which objects are created.
   2. **SuperClass**: the class being inherited from (or a base class or a parent class).
   3. **SubClass**: the class that inherits from another class (or a derived class, extended class, or child class).
      1. The subclass can add its own fields and methods in addition to the superclass fields and methods.
   4. **Reusability**: a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

### Other types of questions asked:

* Share the screen with a code. Find what is wrong in the code or how would you refactor it.
* What would you rate your Java experience out of 10?
* What is your Java level? Do you use it in terms of testing only or in terms of development as well?

## ABSTRACTION

* Abstraction means we **focus on the essential** qualities of something rather than one specific example.
  + Focus on the essential (we know phone need to call(), but we don't know how do iphone, samsung call())
  + Ignore the irrelevant
  + Ignore the unimportant
* In Java, abstraction is achieved **by interfaces and abstract classes.** We can achieve 100% abstraction using interfaces.

## ABSTRACT CLASSES

|  |  |
| --- | --- |
| **CREATING ABSTRACT CLASSES**   * **abstract** keyword is used to create abstract class. * An abstract class **can not** be instantiated. * Goal is to provide reusable variables and methods to sub classes.   **ABSTRACT CLASS RULES REVIEW**   * + It cannot be instantiated directly.   + It may be defined with any number, including zero, of abstract and non­abstract methods.   + It may not be marked as private or final.   + An abstract class that extends another abstract class inherits all of its abstract methods as its own abstract methods.   + The first concrete class that extends an abstract class must provide an implementation for all of the inherited abstract methods | **public abstract class Student{**  **}** |
| **CREATING ABSTRACT METHODS**   * **abstract** keyword is used to create abstract method. * Abstract method **does not have body**, only have signature.   **ABSTRACT METHOD RULES REVIEW**   * + It may only be defined in abstract classes.   + It may not be declared private or final.   + It must not provide a method body / implementation in the abstract class for which it is declared.   + Implementing an abstract method in a subclass follows the same rules for overriding a method. | **public abstract class Student{**  **public abstract void attendClass();**  **}** |
| **CREATING CONCRETE CLASS**   * A subclass of abstract class is called concrete class * A first concrete class must implement all inherited abstract | **public abstract class Student{**  **public abstract void attendClass();**  **}** |
| methods | **public class LocalStudent extends Student{ @Override** |
|  | **public void attendClass(){** |
|  | **System.out.println("attending in person");** |
|  | **}** |
|  | **}** |
| **EXTENDING ANOTHER ABSTRACT CLASS**   * An abstract class can extend another abstract class. If so it is optional to implement abstract methods from abstract super class. * A first concrete class must implement all inherited abstract methods. | **public abstract class LocalStudent extends Student{**  **public void attendClass();**  **}** |

1. **INTERFACE**
   * Contract between a class and outside world
   * Provide set of abstract methods
   * A class implements an interface
   * The class provides the behaviors included in the interface
   * Interface can also store constants

|  |  |
| --- | --- |
| **CREATING AN INTERFACE** | **public interface Teachable{**  **public static final boolean STUDY\_HARD = true; public abstract void canLean();**  **public abstract void doHomework();**  **}** |
| **IMPLEMENTING AN INTERFACE**   * A class can implements more than one interface.   **public class Student implements Teachable, Dreamer{**  ***//implementation code***  **}**   * If a class both extend a class and implement an interface, extends should come first then implements keyword   **public class Student extends Person implements Teachable, Dreamer{**  ***//implementation code***  **}** | **public interface Teachable{**  **public static final boolean STUDY\_HARD = true;**  **public abstract void canLean(); public abstract void doHomework();**  **}**  **public class Student implements Teachable{**  **@Override**  **public void canLean(){**  ***//code***  **}**  **@Override**  **public void doHomework(){**  ***//code***  **}**  **}** |
| **INTERFACE RULES**   * It is a abstract type and can not be instantiated. → * An interface is abstract by default and may not be → marked as final * All fields in interface is automatically public static final even no declared such → | **public interface Teachable{}**  **Teachable t = new Teachable(); *//DOES NOT COMPILE***  **public final interface Teachable{} *//DOES NOT COMPILE***  **public final interface Teachable{ boolean STUDY\_HARD = true; void canLean();**  **public abstract void doHomework();**  **}** |
| **EXTENDING ANOTHER INTERFACE**   * An interface can extends another interface using   **extends** keyword to share functionality.   * Unlike class, an interface can **extends multiple interfaces.** * First concrete class has to implement all the abstract methods from both interface. | **public interface Teachable{ void canLearn();**  **}**  **public interface Mentorship extends Teachable,Bright{ void mentor();**  **}** |