Java Basics

1. What is the difference between a constructor and a method?

A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator.

A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

2. What is the purpose of garbage collection in Java, and when is it used?

The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused.

A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used.

3. Describe synchronization in respect to multithreading.

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources.

Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

4. What is an abstract class?

Abstract class must be extended/subclassed (to be useful). It serves as a template. A class that is abstract may not be instantiated (ie. you may not call its constructor), abstract class may contain static data.

Any class with an abstract method is automatically abstract itself, and must be declared as such. A class may be declared abstract even if it has no abstract methods. This prevents it from being instantiated.

5. What is the difference between an Interface and an Abstract class?

An abstract class can have instance methods that implement a default behavior. An Interface can only declare constants and instance methods, but cannot implement default behavior and all methods are implicitly abstract.

An interface has all public members and no implementation. An abstract class is a class which may have the usual flavors of class members (`private, protected`, etc.), but has some abstract methods.

6. Explain different way of using thread?

The thread could be implemented by using runnable interface or by inheriting from the `Thread` class. The former is more advantageous, 'cause when you are going for multiple inheritance, the only interface can help.

7. What is an Iterator?

Some of the collection classes provide traversal of their contents via a `java.util.Iterator` interface. This interface allows you to walk through a collection of objects, operating on each object in turn.

Remember when using Iterators that they contain a snapshot of the collection at the time the Iterator was obtained; generally it is not advisable to modify the collection itself while traversing an Iterator.

8. State the significance of public, private, protected, default modifiers both singly and in combination and state the effect of package relationships on declared items qualified by these modifiers.

`public:` Public class is visible in other packages, field is visible everywhere (class must be public too)

`private :` Private variables or methods may be used only by an instance of the same class that declares the variable or method, A private feature may only be accessed by the class that owns the feature.

`protected :` Is available to all classes in the same package and also available to all subclasses of the class that owns the protected feature. This access is provided even to subclasses that reside in a different package from the class that owns the protected feature.

What you get by default ie, without any access modifier (ie, public private or protected). It means that it is visible to all within a particular package.

9. What is static in java?

Static means one per class, not one for each object no matter how many instance of a class might exist. This means that you can use them without creating an instance of a class.Static methods are implicitly final, because overriding is done based on the type of the object, and static methods are attached to a class, not an object.
A static method in a superclass can be shadowed by another static method in a subclass, as long as the original method was not declared final. However, you can't override a static method with a nonstatic method. In other words, you can't change a static method into an instance method in a subclass.

10. What is final class?

A `final` class can't be extended ie., final class may not be subclassed. A final method can't be overridden when its class is inherited. You can't change value of a final variable (is a constant).

11. What if the main() method is declared as private?

The program compiles properly but at runtime it will give "`main()` method not public." message.

12. What if the static modifier is removed from the signature of the main() method?

Program compiles. But at runtime throws an error "NoSuchMethodError".

13. What if I write static public void instead of public static void?

Program compiles and runs properly.

14. What if I do not provide the String array as the argument to the method?

Program compiles but throws a runtime error "NoSuchMethodError".

15. What is the first argument of the String array in main() method?

The String array is empty. It does not have any element. This is unlike C/C++ where the first element by default is the program name.

16. If I do not provide any arguments on the command line, then the String array of main() method will be empty or null?

It is empty. But not null.

17. How can one prove that the array is not null but empty using one line of code?

Print `args.length`. It will print 0. That means it is empty. But if it would have been null then it would have thrown a`NullPointerException` on attempting to print `args.length`.

18. What environment variables do I need to set on my machine in order to be able to run Java programs?

`CLASSPATH` and `PATH` are the two variables.

19. Can an application have multiple classes having main() method?

Yes it is possible. While starting the application we mention the class name to be run. The JVM will look for the Main method only in the class whose name you have mentioned.
Hence there is not conflict amongst the multiple classes having `main()` method.

20. Can I have multiple main() methods in the same class?

No the program fails to compile. The compiler says that the `main()` method is already defined in the class.

21. Do I need to import java.lang package any time? Why ?

No. It is by default loaded internally by the JVM.

22. Can I import same package/class twice? Will the JVM load the package twice at runtime?

One can import the same package or same class multiple times. Neither compiler nor JVM complains about it. And the JVM will internally load the class only once no matter how many times you import the same class.

23. What are Checked and UnChecked Exception?

A checked exception is some subclass of Exception (or Exception itself), excluding class RuntimeException and its subclasses. Making an exception checked forces client programmers to deal with the possibility that the exception will be thrown.

Example: `IOException` thrown by `java.io.FileInputStream`'s `read()` method·

Unchecked exceptions are RuntimeException and any of its subclasses. Class Error and its subclasses also are unchecked. With an unchecked exception, however, the compiler doesn't force client programmers either to catch the exception or declare it in a throws clause. In fact, client programmers may not even know that the exception could be thrown.

Example: `StringIndexOutOfBoundsException` thrown by String's `charAt()` method· Checked exceptions must be caught at compile time. Runtime exceptions do not need to be. Errors often cannot be.

24. What is Overriding?

When a class defines a method using the same name, return type, and arguments as a method in its superclass, the method in the class overrides the method in the superclass.

When the method is invoked for an object of the class, it is the new definition of the method that is called, and not the method definition from superclass. Methods may be overridden to be more public, not more private.

25. Are the imports checked for validity at compile time? Example: will the code containing an import such as java.lang.ABCD compile?

Yes the imports are checked for the semantic validity at compile time. The code containing above line of import will not compile. It will throw an error saying, can not resolve symbol

symbol : `class ABCD`

location: `package io`

```
import java.io.ABCD;
```