Java Quick Revision Notes

File Structure

- Each source file can contain only one public class or interface. Two public classes in the same file are not allowed.
- A source file can contain any number of non-public classes. Each class will be compiled into a separate .class file.
- The source file name must be the name of the public class or the interface.
- The source file structure should be as follows:
  Package declaration
  Import statement
  Class or interface definition
- A package statement must appear as the first statement of a source file followed by import statements and class or interface definition.
- Packages are a collection of related classes and interfaces.
- If a class or interface is without a package name, then those are put into a default no-name package.
- java.lang package is imported by default, we don't have to import it explicitly.

main method

- All java application programs must have a main() method.
- main method() is the application's entry point, which would be called by the JVM.
- main method()'s return type is void, access specifier is public, argument is String array and it is static.
- order of public and static does not matter but void should always come before main().
- main() can be declared as final.
- Applets need not have a main() method. It's not a problem even if they have a main method.

Primitives

- There are 8 primitive data types in java. They are not objects.
- boolean is 8 bits in size and stores true or false. It's default value is false.
- byte is 8 bit signed integer and stores values in the range of -2^7 to $2^7 -1$.It's default value is 0.
- char is 16 bit unsigned integer and stores values in the range of 0 to $2^{16}-1$ ie between '\u0000' to '\uffff'. It's default value is '\u0000'.
- short is 16 bit signed integer and stores values in the range of -2^{15} to $2^{15} -1$. It's default value is 0.
- int is 32 bit signed integer and stores values in the range of -2^{31} to $2^{31} -1$. It's default value is 0.
- long is 64 bit signed integer and stores values in the range of -2^{63} to $2^{63} -1$. It's default value is 0l.
- float stores 32 bit floating point values. It's default value is 0.0f.
- double stores 64 bit floating point values. It's default value is 0.0d.

Variables

- Variables declared at class level are called as field or instance variables.
- Variables declared inside a method are called as local variables or automatic variables.
- Field variables are automatically intialized to their default values.
- Local variables are not initialized automatically. They have to be initialzed before using them.
- Arrays, whether local or class-level, are always initialized.

Operators

- The modoulo (%) operator gives the remainder as result, when one operand is divided by another.
- When a int is divided by zero, java.lang.ArithmeticException will be thrown.
- == and != operators are used to check, if the objects are of the same instance.
- equals() method of the object class is used to check, if two objects have the same value.
- instanceof operator is used to determine, if an object reference on the left hand side is an instance of the class or an interface on the right operand.

Arrays

- Arrays are objects.
- The following are legal declarations of an array.

  int i[];

  int[] i;

  int i [][];

  int[] x[];
- The following are the two ways to create an array.

  int i[] = {1,2,3,4,5};

  String str[] = new String[4];
- Array indexes start at 0. str.length for the above statement will result as 5. length is a property of array and is not a method.
- primitives are passed by value and objects are passed by reference. Since arrays are objects, they are always passed by reference.

Access Modifiers

- public access modifier allows access outside of the members package. A class,constructor,method and variables can be declared as public.
- private access modifier allows access only from within the class. A constructor,method and variables can be declared as private. A class cannot be declared as private.
- protected members can be accessed only by classes within the same package and its subclasses in any package. A constructor,method and variables can be declared as protected. A class cannot be declared as protected.
- When no access specifiers are declared, the members can only be accessed within the same package. A class,constructor,method and variables can be declared without any access.
- The accessibilty hierarchy from most restrictive to least restrictive is as follows :
  private -> package(default) -> protected -> public
- Access specifiers are used to provide encapsulation.

Static Modifier

- static access modifier is applicable to a class a whole and not to any particular instance.
- A method or a variable or a block of code can be declared as static.
- When a variable is declared as static, there is only one copy for all instances of the class. If any one of the instance change the value, the change gets reflected to other instances.
- When a method is declared as static, it can be called without creating an instance of the class.
- A static method should not refer to instance variables without using an instance of the class. 'this' keyword should not be used when referring to the instance.
- Static block gets executed when the class is loaded. Static block gets executed even before instance creation.

Final Modifier

- classes,methods and variables can be declared as final.
- A class declared as final cannot be inherited.
- A method declared as final cannot be overridden.
- A variable declared as final may not be changed.

Abstract modifier

- abstract modifier is used to defer implementation to the subclass i.e to provide abstraction.
- classes and methods can be defined as abstract.
- abstract classes cannot be instantiated.
- abstract method doesn't have a body and it is replaced with a semicolon.
- abstract class may or may not have abstract methods but if a class has one abstract method then the class should be abstract.
- abstract class can have non abstract methods.
- abstract without being inherited is of no use. Hence an abstract class cannot be declared as final.

Synchronized Modifier

- methods and blocks of code can be declared as synchronized.
- synchronized methods can be accessed by only one thread concurrently for an instance.
- A class level lock has to be acquired to execute a synchronised static method.

Transient Modifier

- only field(instance) variables can be declared as transient. Local variables cannot be declared as transient.
- transient variables may not be serialized.

Native Modifier

- only methods can be declared as native.
- native modifier indicates method accessing code written in other programming language.

Volatile Modifier

- only field(instance) variables can be declared as volatile. Local variables cannot be declared as volatile.
- volatile variables cannot be declared as final.
- volatile variables may be modified asynchronously by concurrently running threads.

Flow Control

- if..else and switch..case are conditional statements.
- for,while and do..while are iterative or looping statements.
- The following are examples of endless loops:
  for(; ;){}
  while(true){}
- do..while(condition) loop will get executed atleast once. while(condition) loop will only get executed on the condition being true.
- break statement and default condition in switch statement are optional.
- break statement inside a loop causes the control to come out of the loop.
- continue statement inside a loop causes the control to continue with the next iteration.
- There is no goto statement in java.
- Unreachable code results in compilation error. The following is an example:

```
while(false ){
Sytem.out.println("unreachable code");
}
```

Constructors

- Constructors are used for initializing an instance.
- Constructor name should be the same as the class name.
- Constructor's doesn't have a return type.
- Constructors are not inherited.
- Constructors can be overaloaded.
- When no constructors are specified, java provides a no argument constructor by default.
- this() is used to access overloaded constructors and super() is used to access parent-class constructors.
- this() or super(), if used,should be the first line of the constructor code block.
- this() and super() cannot be used at the same time in the constructor.

Inheritance,Overloading,Overriding

- In Java, Polymorphism is achieved using inheritance,overloading and overriding.
- Java supports only single inheritance.
- extends keyword is used for inheritance.
- A class cannot extend more than one class.
- A subclass inherits only the non-private members of the parent class.
- Methods in the same class with same name but different arguements and return type is said to be overloaded. A method is not overloaded, if the return type alone is different.
- Methods and constructors can be overloaded. Overloading is used for code simplicity and runtime polymorphism.
- Only methods can be overridden. Overriding is used to provide a different method behavior in the subclass.
- Overriding method in the subclass must have same signature and return type as the overriden method of the super class.
- Overriding method cannot reduce the access visibility (private is most restrictive) but it can increase the access visibility (public is least restrictive).
- Overriding method can only throw exceptions that the overridden method in the super class throws or it may not throw any exception at all. But it cannot throw new exceptions from different class hierarchy.

Interfaces

- Interfaces along with abstract classes provide abstraction in java.
- Interfaces are used to define abstract methods and constant(static final) variables.
- Classes use implements keyword to make use of interfaces. A class can implement any number of interfaces.
- An interface can extend more than one interface. But the same is not true for classes.
- A class implementing a interface should provide implementation for all the methods declared in the interface. If it doesn't, the class becomes abstract.
- Variable declartion inside a interface can be public or package access and/or abstract.
- Methods declartion inside a interface can be public or package access and/or static/final.

Exception Handling

- java.lang.Exception is the base class of all exceptions.
- java.lang.Throwable is the super class of Exception and Error classes.
- Java uses try..catch..finally constructs to handle exceptions.
- finally block will be executed whether or not an exception is thrown.
- finally block can exist with a try block without a catch block. The exception blocks can be in any of the following combinations - try{}catch(){} or try{}catch(){}finally{} or try{}finally{}.
- Checked exceptions and unchecked exceptions are the ways of classifying exceptions.
- Checked exceptions needs to be explicitly handled using throws clause or catch block.
- Unchecked exceptions are runtime exceptions ie subclass of java.lang.RuntimeException and need not be explicitly handled.
- throw is used to explicitly raise a exception within the program.
- throws clause is used to indicate the exceptions which are not handled by the method. If multiple exceptions are not handled, then they are separated by a comma.
- An overriding method in a subclass must only throw exceptions declared in the parent class or children of the exceptions declared in the parent class or the overriding method may not throw any exception at all.
- The subclass exception should precede the base class exception when used within the catch clause.
- System.exit() will abruptly terminate the JVM. Hence if there is a System.exit() statement in the try/catch/finally blocks, finally will not complete.

Garbage Collection

- In Java, Garbage Collection is automatic. No explicit coding (like destructors) is required.
- Garbage Collector Thread runs as a low priority daemon thread freeing memory.
- An Object is eligble for garbage collection, when there are no references to the object.
- System.gc() and Runtime.gc() methods inform the JVM to run garbage collection but this is only a request to the JVM and it is up to the JVM to run GC immediately or run it when it gets time. Garbage collection can't be forced.
- finalize() method is called by the GC just before releasing the object's memory. It is advised to do a final cleanup of the object in finalize() method.
- When an object is not required, its reference may be nullified which increases the likelihood of garbage collection.

Threads

- A thread is a single sequential flow of control within a program.
- The following are the two ways to create a new thread:
  - Extend the Thread class and override the run() method.
  - Implement the Runnable interface and implement the run() method.
- The following are the states of a thread :
  Ready - instance created but not yet started.
  Running - currently executing.
  Waiting - currently not executing.
  Dead - completed processing.
- start() method puts the thread in ready state and makes the thread eligible to run. start() method automatically calls the run() method.
- wait(), notify(), and notifyAll() are methods defined in the Object class. These methods throw InterruptedException.
- wait() method releases CPU, releases object's lock and puts the thread into pool of waiting threads.
- notify() method moves a thread out of the waiting pool to ready state, but there is no guaranty which thread will be moved out of the pool.
- notifyAll() method moves all waiting threads from the waiting pool to ready state.
- A Thread's priority will be same as the thread which started it. setPriority(int priority) is used to change the priority of a thread.
- yield() is a static method of the thread class which puts currently running thread in to ready state.
- sleep() is a static method of the thread class which puts the current running thread to its waiting state. sleep() throws InterruptedException at runtime.
- setDaemon() method is used to make a thread as daemon thread. Daemon thread is a low priority thread which runs in the background.
- A thread dies after the completetion of run() method. A dead thread cannot not be restarted.
- methods and blocks of code can be declared as synchronized.
- synchronized methods can be accessed by only one thread concurrently for an instance.
- A class level lock has to be acquired to execute a synchronized static method.

Java™
25 Years

Thank you!