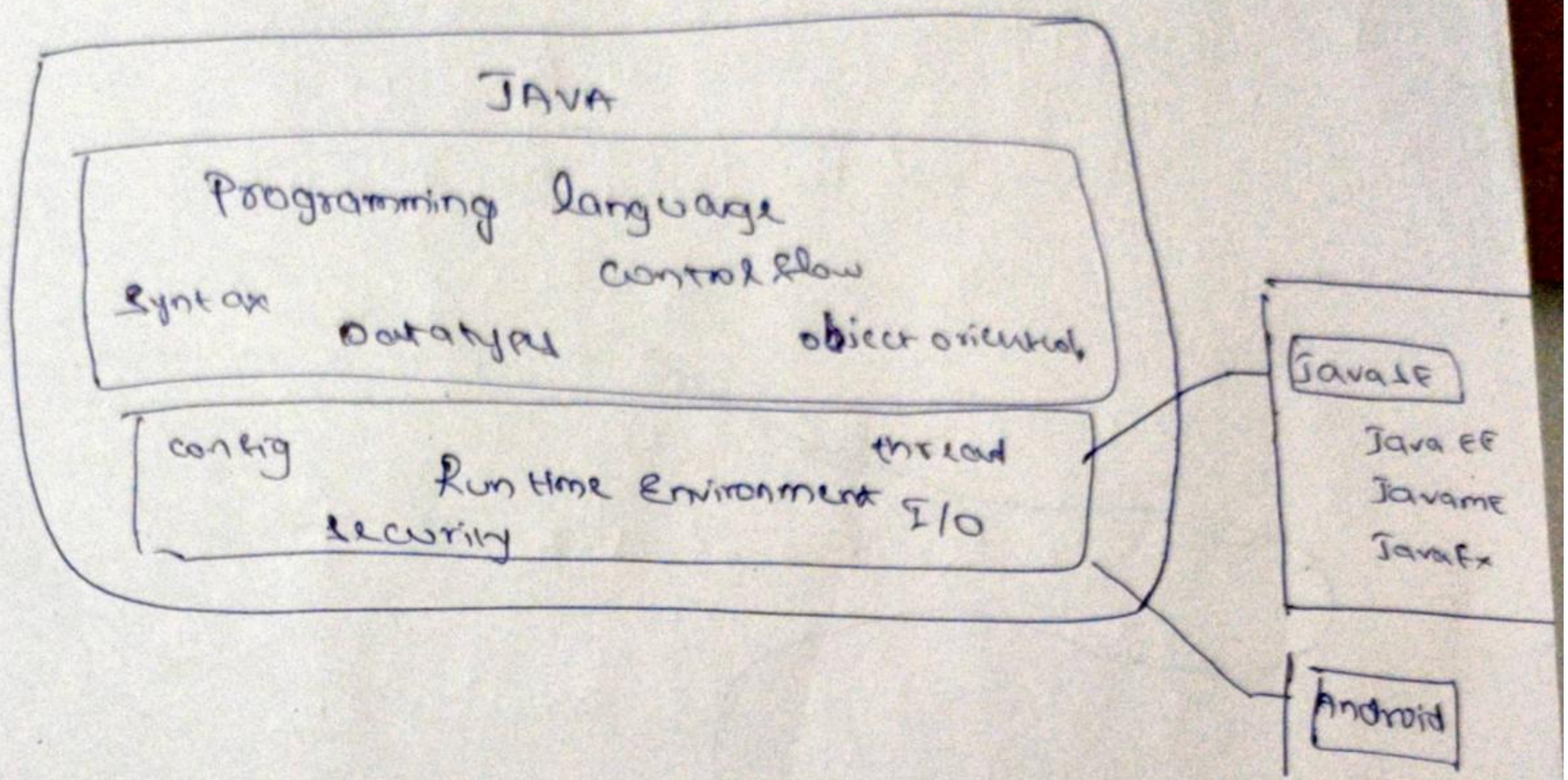
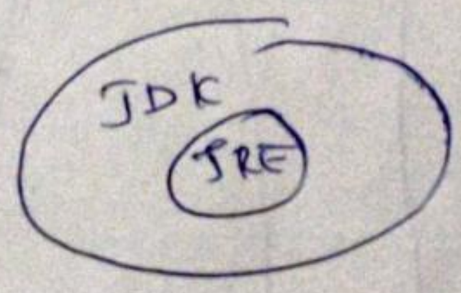
JAVA

What is Java?



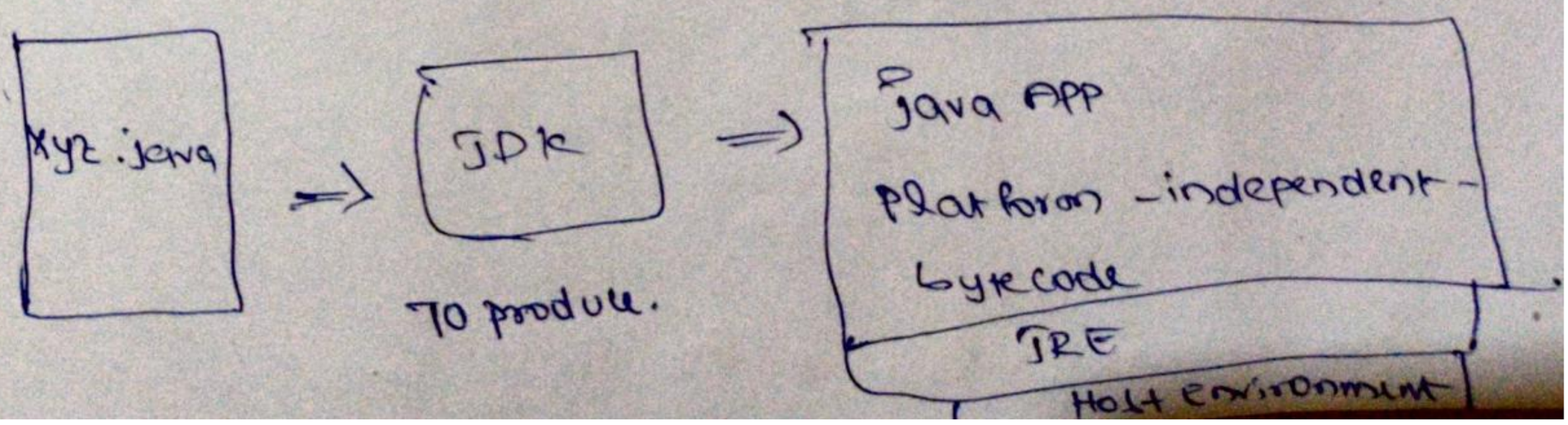
JRE (Java Runtime Envi)

1. Required to Run Java apps
2. End users normally require only the JRE
3. 

JDK (Java Development Kit)

* Provide tools Required to create Java apps
* Developers normally use JDK.
* JDK installation includes JRE

CREATing and Running Java apps



Scanned by TapScanner

- Java is a language and a Runtime Environ
* Specific Environment features may vary (Java SE / ME / EE, JAVA FX, ANDROID)
* Language remains pretty Consistent.
* End users require the JRE
* Developers Require JDK
* many IDE's are available

Main is the Entry Point in Java applications.

Statement Structure

* Statement :- programs made up of Statements
* programs made up of Statements
* Statement Ends with Semicolon;
* Parts Seperated by zero or more white space.

```
System ("Hello ");
System ("Hello");
```

space
tab
newline.

Java gives a lot of flexibility with white spaces

Scanned by TapScanner

* white spaces are just separators.

Statement ends with semicolumn.

Java Comments:-

using comments:-

* Add human readable notes to source code.
* "Hide" source code without deleting.

Types of comments:-

* Single line //
* Block comments /* ---- */
* Java Doc comments /** --- */

Packages

* Packages provides an organization
* follow standard naming convention.
* It Affect our source code file structure.

Package naming Conventions:-

* All lower case	Use reserved domain name to assure global uniqueness	Add further qualifiers to assure uniquen within a compa group.

Package com.cg.training;

Scanned by TapScanner

Package Becomes part of program. and

name.

Class name is unique if its fully qualified

by a package.

Package Name and Source file structure

* Java Requires no correlation Between package names and source file structure.
* But most IDE's require a sub-folder for each part of the package name.

```
src
└ com
  └ pluralsight.
    └ example
      │
      main.java
```

* Execute programs from the command line with the Java Command.
  - Remember to use full class name including the package name.
  - On windows must include JRE Bin folder path environment variable.

Scanned by TapScanner

Variable, datatype, math operator

Variables:

Named data storage	Strongly typed	Value can be modified
	int data; data=100	

* Variable naming is based on a combination of rules and conventions
  - Rules allow the use of letters and numbers are used.
  - Rules allow the used letters, & and _
  - By convention only letters and numbers are used.
  - Rules Requires that first character is not a number
  - By convention it always a letter.
  - By convention follow the style often referred to as "Camel Case"
  - first letter is lower case
  - Start of each word after the first is upper
  - All other letters are lower case.

for ex:- int bankAccountTraining;

Scanned by TapScanner

Primitive Data types:-

Built into the lang	foundation of all other types	four categories of primitive types
		* Integer * floating point * character * Boolean.

Integer:- byte, short, int, long (Integers)

floating points:- float, double (fractional points)

characters :- char type stores a single unicode character.

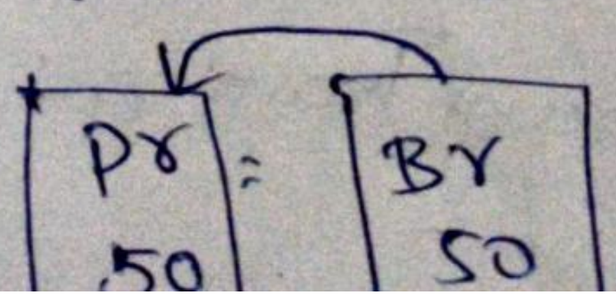
- literal values placed between single quotes.
- for unicode points, use \u followed by hexcode.

Primitive types are stored by value:-

~~Eg~~

int pr = 100

int Br = 50 => Br = 100



Scanned by TapScanner

Prefix operators

++ increments by 1

-- decrements by 1

Prefix

```
int myval = 5
  sysO (++ myval);  6
  sysO (myval);  6.
```

Post fix

```
int  myval = 5
     sysO (myval ++)   5
     sysO (myval)      6
```

Compound assignment operator :-

```
int myval = 50;
myval -= 5
sysO (myval) : (45)
```

* Combines an operation & Assignm
* Applies result of Rig side to left side. Stores the result in variable on left side

Operator precedence :-

operators evaluated in defined order.

Post fix	x++ . x---
prefix.	++x , --x
multiplicative	* , / , %
Additive .	+ -

Scanned by TapScanner

* Operator "=" precedence evaluated from left to Right

* Can overrise precedence with parenthesis. nested parenthis is evaluated from the inside out.

Type conversion :-

* implicit type conversion.

Conversion performed automatically By Compiler.

(widening)

```
int ival = 50
long lval = ival;
```

* Explicit type conversion :-

```
long lval=50.
int ival= (int) lval ;
```

Scanned by TapScanner

Conditional logic

Adding Conditional logic:-

* Relational operators (>, <, >=, <=, ==, !=)
* Conditional assignments
* The if statements
* Logical operator.

Conditional Assignments

```
result = Condition ? true-value : falsevalue;
```

if-else

```
if (Condition)
    true statement;
else
    false statement.
```

Block Statement:-

A Block Statement groups statements into a compound statement.

```
{
    stat1;
    stat2;
}
```

Scanned by TapScanner

Block Statement and variable Scope

* A variable declared inside the block is not visible outside the block.
- A variable's range of visibility is known as the variables scope

Logical Operators

&&, ||, ^, |

loops :-

While loop :- Repeatedly executes the statement as long as the condition is true.

Condition checked at loop first.

do while :- ~~[illegible]~~ Statement in the do execute at least one time if condition is false.

for loop :- Repeatedly Executes a statement as long as the condition is true.

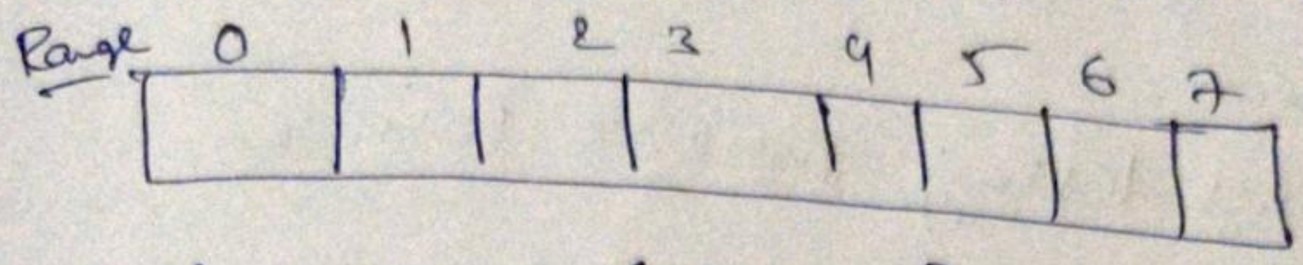
- Condition checked at loop start
- provide simplified notation for loop control values.

```
for (i=1 ; i<n; i++)
        { sysO("Hello)
```

Scanned by TapScanner

Arrays

* Provides an ordered collection of elements



Number of Elements can be found via array's length value.

Ex:-

```
float [] thevals = new float [3];

the vals [0] = 10.0f
the vals [1] = 20.0f
the vals [2] = 15.0f

or

float the vals [] = { 10.0f, 20.0f, 15.0f }

float sum = 0.0f;

for (int i=0; i<the vals.length; i++)
    sum += the vals [i];

System (sum);
```

---

for each loop:-

* Executes once statement for each member in array

```
for (loop variable -declaration : array)
    statement.

float [] thevals = { 10.0f, 20.0f, 15.0f };

float sum = 0.0f;

float (float curval : thevals)

sum = sum + curval;
```

Scanned by TapScanner

Switch

* Transfer control to a statement based on a value.
  - Simplifies testing multiple possible matches.
  - only primitive types supported like char and integers
  - A match can execute more than one statement.
  - Can optionally include default to handle any unmatched values.

```
switch (Testvalue)
 {
 Case value-1 :
     statements.
     :
 Case value-2
     statements
     :
 Case value-n
     statements
 default :
     statements
 }
```

```
int val =10;
switch(ival % 2){
  Case 0 :
  sysO (ival);
  sysO ("iseven);
```

break

Is used to break statement by statement.

Scanned by TapScanner

Classes:- Java is an object oriented programing language.

* object Encapsulates data, operation and usage Semantics.
* Allow Storage and manipulation details to be hidden.
* classes provide a structure for describing and creating objects.
* class is a template for creating an object.

same as filename.

```
class A {
    fields
    methods
    constructors
}
```

Using classes:-

use the new keyword to create a class instance.

```
Flight F1 = new Flight();
```

Applying Access modifiers

Public, private, protected.

Classe names follows same rules as variable na

Class name conventions are similar to variable

* use only letters and numbers.
* first character is always a letter
* Pascal case.
* Simple descriptive nouns.

Method Basic

* Executable code that manipulates state and perform operations.
  - Name
  - Return type.
  - Type of parameter list
  - Body contain with brackets.

```
return type name (type - parameter list){
        statements.
}
```

~~Public class~~

```
Public class my Class {

    Public void ShowSum (float x, float y, int co
        float sum=x+y;
        for (int i=0; i<count; i++)
            Syso( sum);
    }
}
```

Scanned by TapScanner

Exiting methods

* when End of the method is reached.
* A return statement is Encountered.
* An error occurs.
* unless there's an error occur, control returns to the method called.

Method return values :-

* Returns a single value
  - A primitive value.
  - reference of an obj
  - reference of an array.

    Arrays are object.

References this and null

* this Refers to current object
* Allows an object to pass itself as a parameter.
* Null is a reference literal.
  - Represents an uncreated objects
  - Can be assigned to any Reference variable.

fields Encapsulation :-

helps to hide implement - ation details	use method to access control fields.

Scanned by TapScanner

Accessors and mutators

Getters and Setters.

```
Public int get seat ( ) {
        return seats;
        }

Public void Set seats (int seats)
  {
    this . seats = seats;
  }
```

Establishing initial states

Often the default state Established java is not enough.

The object may need to set values or Execute code.

3 mechanisms

- field initialized,
- Constructors
- initialization Blocks.

A fields initial state is Established as Part of object Construction fields receive a "zero" value by default.

Scanned by TapScanner

Field Initializers

* Allow you to specify fields initial value as part of its declaration.
  * Can be simple assignment.
  * Expression.
  * Reference other fields
  * method Call.

"Constructor"

* Executable code used during object creation to set the initial state.
  - Have no returntype.
  - Every class has at least one Constructor.
  - Java provides default Constructors.

Chaining Constructors

* we can chain constructors to reduce the redundancy
* we can add access specifiers to control the constructor visibility.

Initialization Blocks

Parameters passed by making a copy of the value

known as passing "by-value"

* Changes made passed value not to visible outside.

Scanned by TapScanner

Overloading:-

* Same name different parameters
  * Constructor overloading
  * Method Overloading.

Name	Number of parameters	Type of each parameter.

Variable number of parameters:-

Place an Ellipse after a perimeter (...)

Summary

* Parameters are immutable.
* A class may have multiple version of its constructors or methods.
* A method can be declared to accept varying number of parameter values.

Class Inheritance

* Class can inherit from another class.
* Uses Extends keyword.
* Derived class has characteristics of base class.
* hiding fields are not good.

Scanned by TapScanner

Object class

~~object class is the hierarchy of java~~

object class is the root of java class hierarchy.

Useful for declaring variables, fields, parameters that can reference any class or any instance	Defines a number of methods that are inherited by all objects.

Object class methods.

Clone → Create new object instance and duplicates

hashcode → Get a hashcode for the current instance.

getClass → Return type information for the current instance.

finalize → handles special cleanup events

toString → Return string of characters representing the current instance

equals → compare another object to the current instance for equality

Equality

== it Equals the reference.

Scanned by TapScanner

Super

* Similar to this super is an implicit reference to the current object
* Super treats the object as if it is an instance of its base class.

final

Public final class pass {

}

Abstract

* If you have abstract method then class must be abstract.
* It's possible to have abstract class without abstract methods.

Inheritance and Constructors

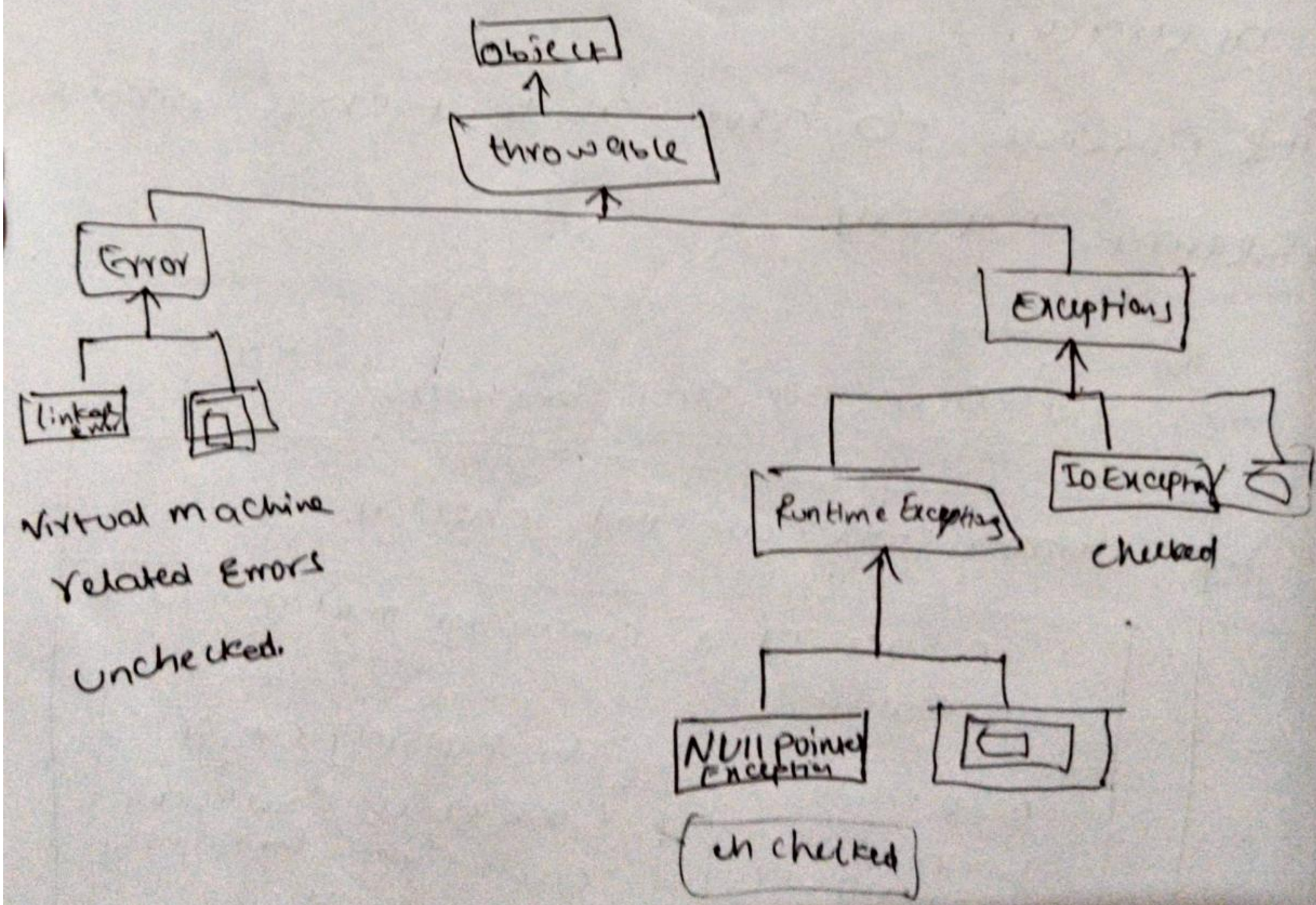
* Constructors are not inherited.
  - A base class constructor must always be called
    - By default base class no-argument constructor is called.
    - Can explicitly call a base class constructor using super followed by parameter list.
      - must be first line of constructor.

Scanned by TapScanner

Exceptions

* Exception Can be thrown ~~at~~ when the Error occurs. It interrupts the normal flow of Execution. It can be recovered.

```
try {
    Code
  }
 catch {
    Error msg
  }
 finally {
    connection close
  }
```



Scanned by TapScanner

Exception can be handled by type.

Each exception type can have a seperate catch block	Each catch is tested in order from top to bottom	First assignable catch is selected.

Start catch blocks with most specific exception types.

Exceptions and methods

Exceptions propogate up the call stack. can across the method boundaries

Exceptions are part of methods contract method is responsible for any checked exception that might occur.

Catch the Exception	Document that the exception might occur

Throwing Exceptions

You can create your own Exceptions called as custom exception type

Normally inherit directly from Exception class, make them checked Exceptions,

Constructors are often their only members.

Most reloired functionality is inherited.

Scanned by TapScanner

Working with packages

A package is a group of related types.

Create a namespace	provide an access boundary	Act as a unit of distribution

Packages create a name space in order to avoid collision of names

Determining a type's package

Type imports

Type imports guide compiler to fully qualified names.

Single type import	import on demand.

Limiting access to package contents

No modifier, public, private protected.

Packages provide a unit of Distribution.

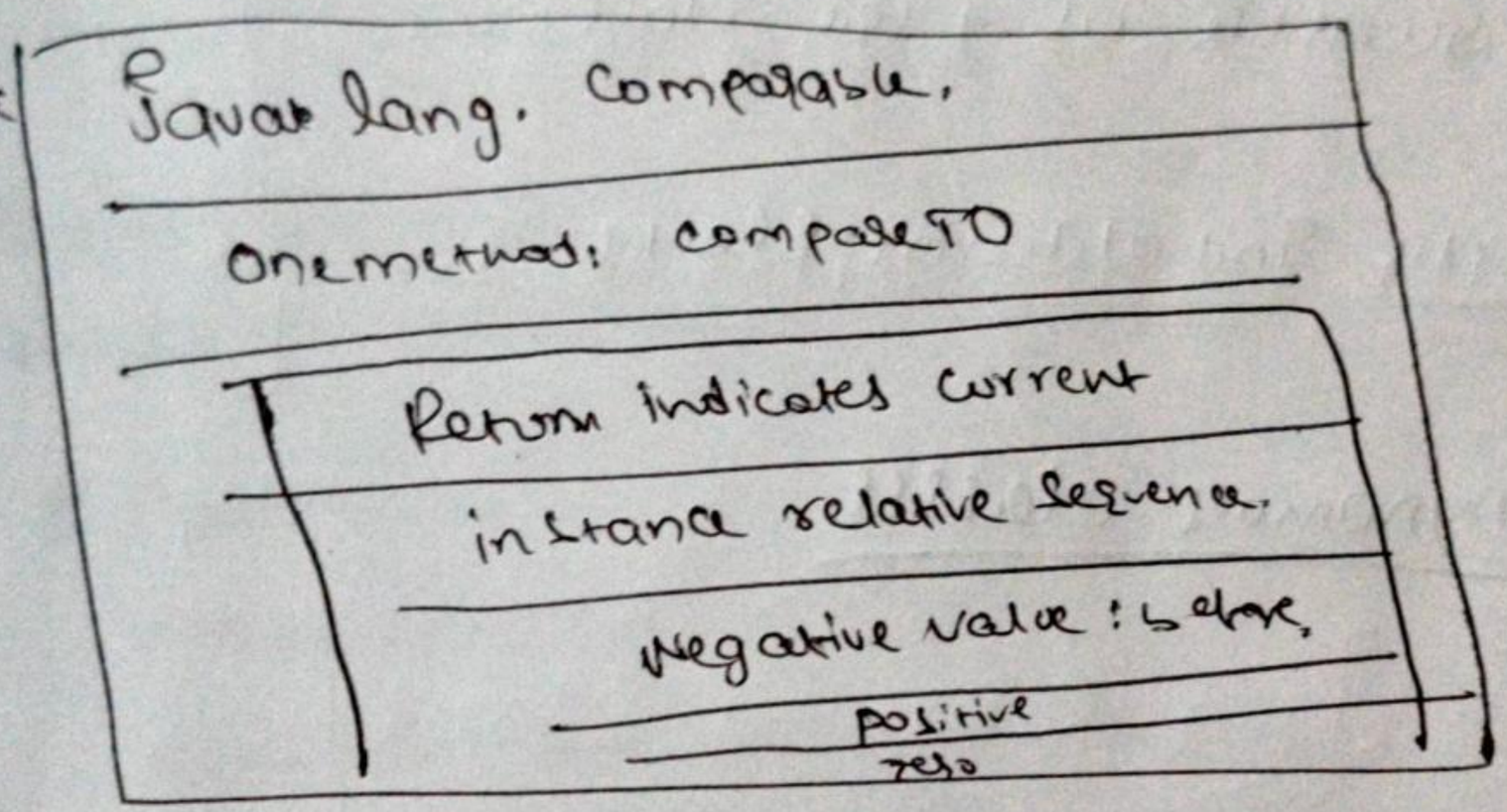
Packages provide a predictable software structure.

Scanned by TapScanner

Archive files : places package folders structure into a single file.

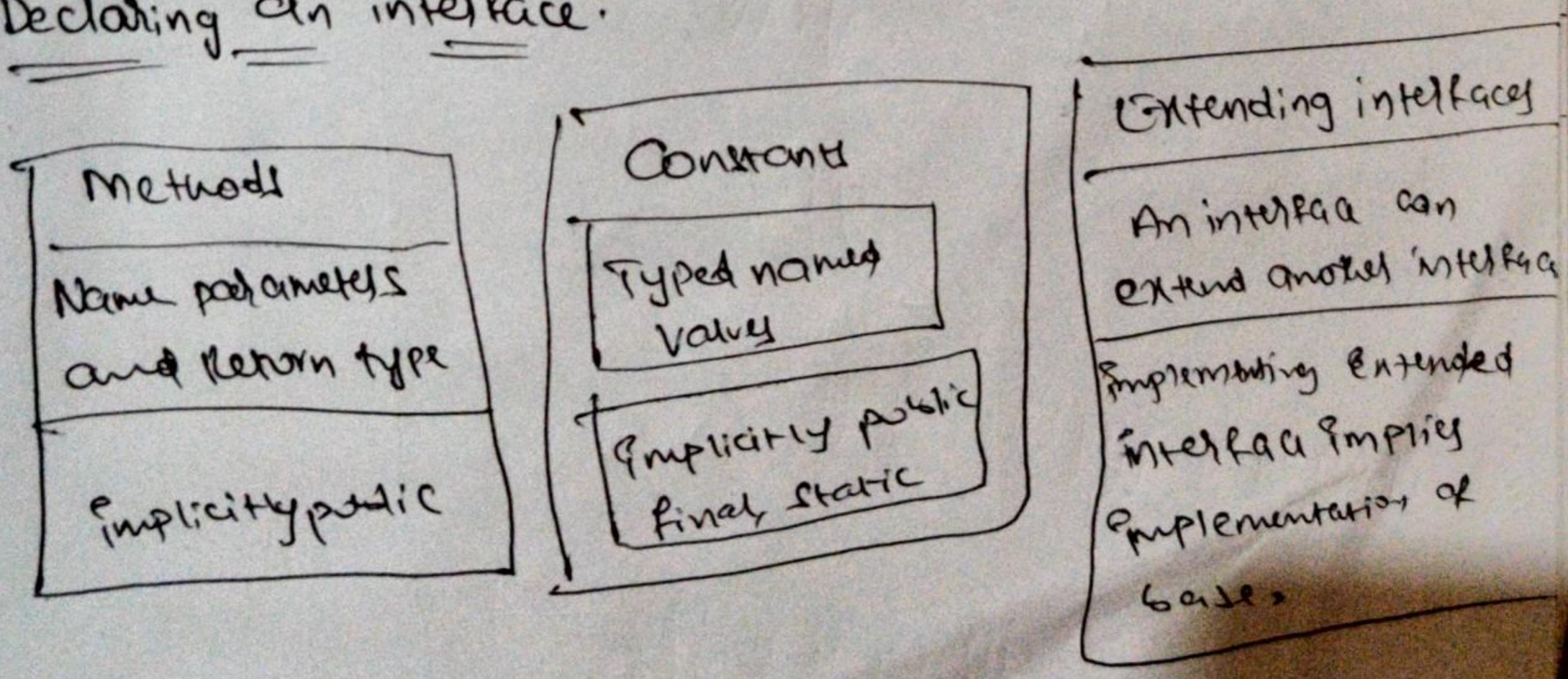
Creating Abstract Relationship with Interfaces

* Provides Contract, No implementation.
* Interfaces donot limit Classes.



what is an interface?

Declaring an interface:



Scanned by TapScanner

Static members, Nested types

Static members

Static members are shared class-wide

Declared using static keyword

Accessible using the class name.

Static Initialization Blocks

Anynomous classes

Scanned by TapScanner

Memory management

why GC?

forms of GC

- Do Nothing
- Reference counting
- mark and Sweeping.
- Copying
- Generational
- incremental.

Major gc → slow → Tenured space is full → collects old and young

Minor gc → fast

- G Cardtable,

  parallel collector

  serial collector

  parallel old collector.

  Mark and sweep collector.

G1 Collector :- Supported in java 7

is a Compacting Collector.

Planned a Replacement for CMS

Meant for server application.

Breaks heap into Regions,

Objects are Evacuated b/w spaces.

Scanned by TapScanner

Which collector?

MXBeans There is a Java mx Beans for the garbage collector.

- Name of Collector
- Number of collections
- Time of Collections.

Jstat :- Command line tool

jstat -option <pid> <interval> <count>

Visualgc :-

Java References classes

Java has always strong references

- object not gc'd until references are released,

Reference Rules

Strong → soft → weak → phantom

Soft will be collected only if there is a memory pressure

Weak will be collected immediately.

Phantom references different to the other two

Scanned by TapScanner

How Java Reference.

Usage of reference types :-

Weak reference :- Associate meta data with another type

- use weak hash map

Soft Reference

- can be used for caching.

Phantom reference.

- interaction with the garbage collector.

Using Reference Queue :-

Pass a reference queue to constructor when creating the reference object.

- optional
- Except for phantom reference.

Reference types enqueued to Reference queue when all strong References cleared

- Reference object is added to the Reference queue
- Reference queue has poll and remove methods
- poll returns immediately
- a timeout.

Scanned by TapScanner

Can be used to attach clean up code.

- Extend reference type.

Phantom

Used instead of finalizers

finalizers can have issues.

Scanned by TapScanner