Java Rules

1) Two public class cannot be in the same source file and that filename should be same as the public class name.

2) package statement should be 1st statement in the sourcefile followed by import statement

3) Singleton class → limit no of object to 1. Use private constructor & static getInstance() method, use lazy instantiation to ensure instance is created only when needed.

4) To declare long use 12L
float use 12.3f

5) Java literal decimal = 100
octal = 0144; (V-Vimp)
hexa = 0x64;

6) Unicode char = '\u0001';
String = "\u0001";

7) Using char x = 121 → Ascii
print(x) · print(x+2)
o/p = y o/p = 123

~~char x = '\u0001'~~

char x = '\u0021'

print(x)

print(x+3)

as ascii

! = 33

In java '5' + 3

↳ convertible ascii int

'5' → ch in char.

8)

declaration datatype name.

instantiation new Constructor();

initializer name = ; → using equal.

9) Local variable don't have access modifier as they only work inside function & are destroyed if we exit the fun/method. They have no default value

10) Local variable are implemented at stack area.

11) Instance variable in heap.

12) Class/static variable: Can only have 1 copy per class no matter how many objs are created.

13) public static + final → constant.

14) created when program starts.

15) modifier modifies the meaning of class, method, variable.

16) protected means all classes in package can access it & all subclasses also can access it.

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
Public	+	+	+	+	+
Protected	+	+	+	+	o
Default	+	+	+	o	o
Private	+	o	o	o	o

+ → accessible

o → not accessible

17) Non Access modifiers

static → ~~class~~ methods & variable
final → class & methods & variable
abstract → classes & methods
syn & volatile → threads.

for interfaces (implicitly)
fields → public, static, final.
methods → public

18) Access modifiers:

private.
→ constructors ~~from~~ used for singleton class.

→ class & interface can't be private

→ "To access private variables if public getter methods are present in class."

→ private used to encapsulate data (field) from outside world (can't set the variable outside class).

~~Task~~
Public class from different package still needs to be imported.
→ All public fields & methods of class are inherited by such class.

→ main method of app has to to be public otherwise it can't be called by a Java interpreter to run the class

→ Protected → package + subclass.
class (variables, methods, constructors)

→ protected can't be applied to class & interface. & also methods & fields in interfaces.

Access control & Inheritance

① methods public in superclass also must be public in all subclass

② methods protected in superclass must be public / protected in subclass.

Imp ③ Methods declared private are not inherited.

9) Non Access Modifier.

Static var → exist independently of any instances created for class.
↓
classVar → only 1 copy of static var exists regardless of no of instances of the class.

Static methods → exist independently of any instances for the class.

→ Static methods can't use non static fields

Final

→ Final variable can be explicitly initialized only once. i.e

not instantiated → reference variable declared final can never be reassigned to diff^n object.

Not Imp → but the data within obj can be changed

So state of obj can change not ref^n of obj

→ This implies that Java has no concept of object immutability.

→ final + static → const^n

Imp → Final methods can't be overridden by subclass.

Imp → Final methods can overridden through inheritance

→ Final class can't be inherited.

Abstract method

method

→ An abstract method is method declared without any implementation

→ method body is provided by subclass.

→ Abstract methods can't be final

→ ends with semicolumn.

public abstract sample();

class

→ If class has 1 or more abstract methods then class has to be declared abstract.

→ Any class that extends an abstract class must implement all the abstract methods of super class, unless the subclass is also an abstract class.

20) Integer.toString (100, 2) → binary representation.

Integer.toString (100, 8) → Octal. representation

" " " (100, 16) → Hex "

21) For negative binary representation, 2's complement → Integer.toBinaryString

22) # Integer is 32-bit signed datatype. Integer.toBinaryString() returns a string representation of the integer argument as unsigned integer base 2.

So, Integer.parseInt(Integer.toBinaryString(x)) will generate exception if x is -ve

Safeway → Integer.toString(x, 2)

23) ~~#~~ A private final instance variable can't be assigned a value inside a method. as a method can be called multiple times. but can be assigned in a constructor.

24) int y = Integer.parseInt(Integer.toString(100, 2)); ↳ convert 100 to binary.

25) ~~va~~ +1 → to represent -ve no.

26) 6 << 1 → 6*2 6 >> 1 → 6/2
6 << 3 → 6*2^3

27) Arithmetic right shift //

msd remain same.

100 1100 >> 4

~~1111~~ 1100 1100
1111 0011
1111 1001
1111 1100

(sign remain same)

28) instanceof → checks whether the object is of particular type
(class/interface)

IS A test.

also checks for subclass

- getclass returns class of object
Class.class → returns class object of class.

to check for only subclass

(obj instanceof Event && object.getClass() != Event.class)

29) Precidence of Java Operators

conditional ?: (right to left)

30) Enhanced for loop

String[] names = {"Jai", "Ajay", ...};

for(String name : names)
{

}

31) Boxing → primitive to object

int to Integer.

32) Math.random() generates random no.

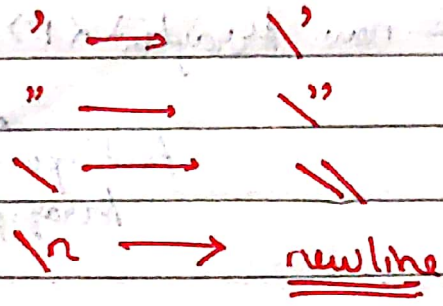
33) In java,

char[] ch = {'a', 'b', 'c', 'd', 'e'};

not

[ ]

34) Escape sequence (Backslash '\')



35) Strings are immutable

36) new String (CharArray) creates string.

37) String methods

- length()
- concat()
- charAt() → int
- compareTo(String2) → lexicography comp
- equalsIgnoreCase
- indexOf() → character
- replace()
- replaceAll()
- split()
- substring()
- toCharArray()
- trim()

38) Convert Arrays to ArrayList.

ArrayList<E> name = new ArrayList<E>(Arrays.asList());

first import Arrays from java.util.

ref ~~name~~ of array

39) Collections.reverse(ArrayList)

40) Scanner.next()
.nextLine()
.nextInt()

41) We can't use Scanner.nextLine() after nextInt() / next().
. If we use nextFoo() → Anything but Line.
& then use nextLine.
this nextLine consumes the /n character which next/nextInt() don't consume.

42) nextLine also consumes /n but rest don't.

43) next() will only return what comes before space.

44) Scanner.useDelimiter is useful for many application like if you have csv & want to print out individually. eg- A, B, C
use useDelimiter(",")

45) You can use regex inside wde limiter.

46) Str.split(@ regex) is used to split using a regex.