Int. Que Ans:-

① final, finally & finalize

final:-

1) final is a modifier applicable for classes, methods and variables. If a class declared as final then we can't extend that class. i.e. we can't create child class for that class.

2) If a method declared as final then we can't override that method in the child class.

3) If a variable declared as final then it will become constant and we can't perform re-assignment for that variable.

finally:-

It is block always associated with try catch to maintain cleanup code.

→ finally meant for cleanup activities related to try block.

finalize():-

It is a method which is always invoked by garbage collector just before destroying an object to perform cleanup activities.

→ finalize() meant for cleanup activities related to objects.

Scanned by CamScanner

2. String , StringBuffer

String:-
It is immutable. (can't change)
Once we creates a String Object we can't perform any Changes in the existing Object.

```
e.g. String s1 = "welcome";
     s1.concat ("Mysuru");
     S.O.P. (s1);    // welcome
```

StringBuffer:-
It is mutable. (Changeable)
Once we creates a StringBuffer Object we can perform any type of changes in the existing Object.

```
e.g.:- StringBuffer s = "welcome";
       s.append ("Mysuru")]
       S.O.P.(s) ; // welcomeMysuru
```

3. (==), equals()
double equal operator (==):-
It is used for reference comparison.

equals() method:-
It is used for content comparison.

4. StringBuffer & StringBuilder:-

StringBuffer	StringBuilder
methods inside - synchronized	Non - synchronized
(only one thread) Thread - safe	Not Thread safe (multiple thread)
Performance is low	Performance is high
introduced 1.0 version	introduced 1.5 version

Scanned by CamScanner

⑤

1. String:- If the content is fixed and won't change frequently then we should go for String.

2. StringBuffer:- If the content is not fixed and keep on changing but Thread safety is required then we should go for StringBuffer.

3. StringBuilder:- If the content is not fixed and keep on changing and thread safety is not required then we should go for StringBuilder.

6) Interface VS Abstract class VS Concrete class

Interface:- If we don't know anything about implementation just we have requirement specification (100% Abstraction) then we should go for interface. Ex:- Servlet

Abstract class:- If we are talking about implementation but not completely (Partial implementation) then we should go for Abstract class. Ex:- GenericServlet & HTTPServlet

Concrete class:- If we are talking about implementation completely and ready to provide service then we should go for concrete class. Ex:- MyOwnServlet

Access Specifiers VS Access Modifiers:-

→ In Java there is no terminology like specifiers, all are by default considered as modifiers only.

Public	default	synchronized	Strictfp (1.2V)
Private	final	abstract	transient
Protected	Static	native	volatile

Scanned by CamScanner

7) Abstract Class vs Interface

Interface	Abstract class
→ If we do not know anything about implementation just we have requirement specification then we should go for interface.	→ If we are talking about implementation, but not completely (partial) then we should go for Abstract class.
→ Every method is always public and abstract.	→ Every method need not be public and Abstract.
→ 100% pure Abstract class.	→ It can take concrete methods.
→ We can't declare interface method with the following modifiers Public :- private, protected, Abstract :- final, static, native	→ There are no restrictions on abstract class method modifiers.
→ Every variable present inside interface is always public, static and final whether we are declaring or not.	→ The variables present inside Abstract class need not be public, static and final.
→ We can not declare interface variables with the following modifiers :- e.g. private, protected, transient, volatile	→ There are no restrictions on Abstract class variable modifiers.
→ For interface variables compulsory we should perform initialization at the time of declaration otherwise compile time error.	→ For Abstract class variables it is not required to perform initialization at the time of declaration.
→ Can not declare instance & static blocks.	→ Can declare instance & static blocks.
→ Can not declare constructors.	→ Can declare constructors, which will be executed at the time of obj creation.

Scanned by CamScanner

8) System.out.println()

```
class System
{
    static PrintStream out;
}
System.out.println("Hello");
```

-> System is class present in java.lang package.
-> 'out' is a static variable present in systems class of type PrintStream.
-> println() is a method present in PrintStream class.

Scanned by CamScanner

9) Public Static void main(String[] args) :-

→ Whether class contains main() method or not and whether main() method is declared according to requirement or not these things won't be checked by compiler. At runtime, JVM is responsible to check these things.
→ At runtime if JVM is unable to find required main() method then we will get runtime exception saying NoSuchMethodError: main

* Public :- To call by JVM from anywhere.
* Static :- without existing object also JVM has to call this method & main method no way related to any object.
* void :- main() method won't return anything to JVM.
* main(String[] args) :- command line arguments.
↳ this is name which is configured inside JVM.

→ Syntax is very strict if we perform any changes we will get runtime exception saying NoSuchMethodError: main
Following changes are acceptable :-
① Order of modifiers :- instead of "Public static" we can "Static Public".

② Declare "String[]" in any form :- main(String[] args)
main(String []args)
main(String args[])
③ Instead of 'args' we can take any vaild java identifier. e.g:- x, y...
④ Replace String[] with var arg parameter :- main(String... args)
⑤ with following modifiers :- final
synchronized - (only one thread)
strictfp.

→ final static synchronized strictfp public void main(String... args)
↳ this will work. jvm can run the main method.

Q. which of the following are valid main method declarations
1. public static void Main(String[] args) (X)
2. public static int main(String[] args) (X)
3. public static void main(String args) (X)
4. public final synchronized strictfp void main(String[] args)(X)
✓5. public static final synchronized strictfp void main(String[] args)
✓6. public static [illegible] void main(String... args)

→ we won't get compile time error any where but at runtime we will get exception in all cases except last two.

Case-1
→ Overloading of main method is possible but JVM will always call String[] argument main method only.
Case-2
→ Inheritance concept applicable for the main method. Hence while executing child class if child class doesn't contain main method then parent class main method will be executed.

ex:-
```
class P
{
    P.S.V.m (String[] args)
    {   S.O.P. ("Parent main");
    }
}

class C extends P
{
}
```

$$\begin{tikzcd}
 & \text{javac P.java} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{P.class} & & \text{C.class}
\end{tikzcd}$$

java P↵
O/P :- Parent main
java C↵
O/P :- Parent main

Scanned by CamScanner

Case 3:-

→ It seems overriding concept applicable for main method but it is not overriding it is method hiding.

e.g.:-
```
class P
{
    P.S.V.m (String[] args)
    {   S.O.P. ("Parent main");
    }
}
class C extends P
{   P.S.V.m (String[] args)
    {
        S.O.P. ("Child main");
    }
}
```
It is method hiding but not overriding

$$\begin{tikzcd}[column sep=small] & \text{javac P.java} \arrow[dl] \arrow[dr] & \\ \text{P.class} & & \text{C.class} \end{tikzcd}$$

java P↵ O/P:- Parent main

java C↵ O/P:- Child main

Note:- For main method inheritance and overloading concepts are applicable.

→ Overriding concept is not applicable instead of overriding method hiding concept is applicable.

Java 1.6 version

→ RE: NoSuchMethodError : main

Java 1.7 version

→ Error: Main method not found in class test, please define main method as public static void main(String[] args).

→ From 1.7 version onwards to run a java program main method is mandatory. Hence, even though class contains static blocks they wont be executed if the class doesn't contain main() method.

→ If the class contains main() method whether it is 1.6 or 1.7 version there is no change in execution sequence.

```
Class Test
{
  Static
   {   S.O.P.("Static block");
   }
   P.S.V.m (String[] args)
   {   S.O.P.("main method");
   }
}
```

O/p

for both 1.6 & 1.7

```
O/p:- Static block
      main method
```

Q. Without writing main method is it possible to print some statements to the console?

Ans:- Yes, we can by ~~s~~ using static block.

But this rule is applicable until 1.6 version only from 1.7 version onwards main() method is mandatory to print some statements to the console.

Scanned by CamScanner