

Assignment - 4

classmate

Date _____
Page _____

1) What does the static keyword mean in java? Explain diff between static & non-static methods.

→ In java static keyword is used to indicate that a member (variable, method, block or nested class) belongs to class itself rather than to instances of the class.

Static

Non Static

1) static method can be accessed without creating instance of class.

To access non-static method we have to create instance of class.

2) You cannot access non-static variables in static methods.

You can access static & nonstatic methods.

3) You can't override static methods.

You can override non-static methods.

4) static is also known as class variable.

Non-static also known as instance.

5) static method can be called without creating instance of the class.

Non-static method must be called on an instance of the class.

6) static method allocate memory only once.

— memory allocation occurs even time instance of class is created.
— Non-static m.

7) static methods store in method area.

Store

2) What is role of static keyword in the context of memory management.

→ static keyword in java plays crucial role in memory management by ensuring that static variables & methods are stored in method area & ~~are~~ loaded only once when class is first loaded.

3) Can static methods be overloaded & overridden in java? How static variables shared across multiple instance of a class?

⇒ Yes, static method can be overloaded by defining multiple methods with the same name but different parameter lists.

No, static methods cannot be overridden in same way as instance methods. Instead, method hiding occurs, meaning the method called based on the reference type at compile-time.

Static variables are shared multiple instances because they belong to class itself, not individual objects.

When class is loaded the static variable is stored in the method area. & all instances of class share same memory location for that variable, ensuring consistency across obj

4) What is significance of final keyword in Java?

- 1) Final variables - When a variable is declared as final, it can be assigned only once. After its initial assignment its value cannot be changed. This is used for constant.

```
final int MAX_VALUE = 100;
```

- 2) Final methods - Declaring method final prevents subclasses from overriding it. This is useful.

- 3) Final class - A class marked as final cannot be subclassed, preventing inheritance.
ex - String class in Java is final.

- 5) ~~to~~ Narrowing Conversion & Widening Conv in Java.

→ Widening Conversion → (Implicit Casting)
when smaller data type automatically converted into larger data type.
It is safe doesn't result in data loss.

Ex - int to long, float to double.

Date _____
Page _____

Narrowing Conversion — explicit casting
happens when larger datatype is explicitly
converted into smaller DT.

Ex — double to long, int.

6) Example of Narrowing & widening
Conv. between primitive DTs.

→ Widening Conversion (automatic) (implicit)

```
int num = 100;  
long bigNum = num; // widening int to long
```

Narrowing Conv (explicit) —

```
double decimalValue = 9.78;  
int intValue = (int) decimalValue // Narrowing
```

7) How does Java handle potential loss of
precision during narrowing Conversion?

→ Java requires explicit casting during narrowing
conversions because it recognizes the potential
for loss of precision. when larger type
is narrowed to smaller type, only lower
bits are retained which may result in

truncation of the value or incorrect results.

ex - `double num = 9.99;`
`int result = (int) num;` // result is 9.

Fractional part is lost when converting from double to int.

8) Explain Concept of automatic widening conversion in Java.

→ widening conversion in java happens automatically when you assign a smaller primitive type to larger one.

- This is because - larger data type can accommodate value of smaller type without any data loss.

Example -

```
int smallNum = 42;  
double bigNum = smallNum;
```

int value 42 is automatically widened to double, which can hold larger values.

Q) What are implications of narrowing & widening conversions on type compatibility & data loss?

→ * Widening Conversions — are safe. —
larger type can always accommodate value of smaller type. They ensure type compatibility without any risk of loss.

* Narrowing Conversion — may lead to data loss. Smaller type cannot always hold the larger value. This is why java requires explicit casting for narrowing conv.

Example —

```
long largeValue = 150L;
int narrowedValue = (int) largeValue;
```