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

=> The ~~key~~ static keyword is used to declare class level variables & methods meaning they are shared amoz all instances of the class. Static variables are stored in the static memory area and only one copy of a static variables exists regardless of how muy instances of the class are created. This helps manage memoy by ensuring that only one instance of the variables is maintained, reducing redundancy.

2) Can static methods be overloaded and overridden in Java?
How static variables shared across multiple instances of a a class?

=> Overloading :- Yes, static methods can be overloaded in Java.
Overloading refers to defining multiple methods with the same name but different parameter lists within the same ~~chess~~ class.

Overriding :- static methods cannot be overridden, while you can define a static method with the same name in

a subclass (which hides the static method in the superclass) this is not considered.

3) What is the significance of the final keyword in Java?

=) The final keyword in Java is used to impose restrictions on variables, methods, and classes.

- Final variables :- If a variable is declared as final its value cannot be changed once it is initialized. It must be initialized during declaration or in a constructor.

- Final methods :- A final method cannot be overridden by subclasses, ensuring that its implementation is preserved across the class hierarchy.

- Final classes :- A class marked as final cannot be ~~executed~~ extended. This is used to prevent inheritance.

#7 What are narrowing and widening conversions in Java?

=> Widening Conversion :-

This happens when a smaller primitive data type is converted to a larger one. It is done automatically by the Java compiler, as there is no risk of data loss. This process is known as implicit casting.

Eg :- Converting an int to a long or a float to a double

Narrowing Conversion :-

This happens when a larger primitive data type is converted to a smaller one.

Narrowing Conversions are not done automatically and explicit casting is required. There is a risk of data loss, as the larger data type may not fit within the range of the smaller type

Ex :- Converting a double to an int.

5. Provide examples of narrowing and widening conversions between primitive data types.

=>
```
int i = 100;          widening
long l = i;
double d = l;
```

Narrowing conversion :-

```
double d = 9.8;
int i = (int) d;

System.out.println(i);
```

6. How does Java handle potential loss of precision during narrowing conversions?

=> Java requires explicit casting when performing narrowing conversions to make the developer aware of potential data loss. This explicit cast signals that the developer is aware of and accepts the risk of truncation or loss of precision.

for eg:-

```
double = 100.04;
int i = (int) d;
```

In this case, d will lose its fractional part when converted to an int and only the integer portion (100) will be stored in i.

7) Explain the concept of automatic widening conversion in Java.

=> Automatic widening conversion happens when a smaller sized primitive type is assigned to a larger sized one without explicit casting. Java allows this because there is no risk of data loss as the larger type can accomodate all value of the smaller type.

```
int   i = 10;
long  l = i;

float f = l;
```

In this eg. the int value 10 is automatically widened to long & then to float without requiring any manual cast.

Implications of narrowing and widening conversions on type compatibility and data loss.

Type Compatibility :-

- Widening Conversions are generally type safe and automatic because smaller types can fit into larger types without the risk of data loss.

- Narrowing Conversions require explicit casting because there is potential for data loss, making them less type safe. The developer has to manually acknowledge & handle the conversion.

② Data loss :-

Widening Conversions do not cause data loss because the destination type has a larger or equal range than the source type.

- Narrowing conversions can lead to data loss (e.g losing fractional values when converting a double to a or int) or Overflow (eg. converting a large long values to a int.