CDAC Mumbai PG-DAC August 24

Assignment No- 4

1) Write a program that demonstrates widening conversion from int to double and prints the result.
Code:

```
package program;

public class Widening {

	public static void main(String[] args) {
	int a =49;
	double d =a;
	System.out.println("int value:" +a);
	System.out.println("widened double value: "+d);
	}

}
```

Output:
int value:49
widened double value: 49.0

2) Create a program that demonstrates narrowing conversion from double to int and prints the result.
Code:

```
package program;

public class Narrowing {

	public static void main(String[] args) {
		double d= 42.55;
		int a = (int)d;
		System.out.println("duble value: "+d);
		System.out.println("narrowing int value: "+a);

	}

}
```

Output:
duble value: 42.55
narrowing int value: 42

3) Write a program that performs arithmetic operations involving different data types (int, double, float) and observes how Java handles widening conversions automatically.
Code:

```
package program;

public class ArithmaticWidening {

	public static void main(String[] args) {
		int a=10;
		double b=5.5;
		float f= 3.2f;

		double result1=a+b;
		float result2=a+f;
		double result3 = b*f;
		double result4=a/b;
```

```
        System.out.println("int + double : "+result1);
        System.out.println("int + float: "+result2);
        System.out.println("double * float: "+result3);
        System.out.println("int / double:"+result4);

    }

}
```

Output:

```
int + double : 15.5
int + float: 13.2
double * float: 17.600000262260437
int / double:1.8181818181818181
```

4) Write a Program that demonstrates widening conversion from int to (double,float, boolean, string) and prints the result.

Code:

```
package program;

public class Widening {

    private static String String;

    public static void main(String[] args) {
    int a =49;
    double d =a;
    float f=a;
    boolean b=a>0;
     String = Integer.toString(a);
    System.out.println("int value:" +a);
    System.out.println("widened double value: "+d);
    System.out.println("float: "+f);
    System.out.println("boolean :"+b);
    System.out.println("String: "+String);
    }

}
```

Output:

```
int value:49
widened double value: 49.0
float: 49.0
boolean :true
String: 49
```

Assignment No :-4

Page No.
Date

Q. what does the Static keyword mean in java? Explain the difference between Static & non-static keyword

ans Static keyword is used to indicate that a particular member (variable or method) belongs to the class itself rather than to instances OF the class.

Static Method :-

- Belong to the class : A static method can be called on the class itself not on instance OF the class
  ex:- you can call ClassName.methodName() without creating an object
- NO Access to instance Variables : Static methods cannot directly access instance variables or instance methods. They can only access static variables or other static methods OF the class
- Common use cases : Static Methods are OFten used for utility functions, such as mathematical calculations or factory methods that return instance OF the class

Non-Static methods :-

- Belong to instance : Non-static methodr can only be called on an instance OF the class for ex: you must create an object to ~~create on~~ call objectName.methodName()
- Access to instance Variables: Non-static Method can access both instance variables and static variables/methods. They can operate on the specific data OF the object they are called on.
- Common use cases : Non-static methods are used when the behavior or data is specific to a particular instance of the class

Q.1 What is the role of Static keyword in the context of memory management.

The Static Keyword in Java is mainly used for memory management. The Static keyword in Java is used to share the same variable or method of a given class. The users can apply Static keywords with variables, methods, blocks, and nested classes. The Static keyword belongs to the class rather than an instance of the class.
- The Static keyword is a non-access modifier in Java.
- Characteristics of Static keyword :-
  - Shared memory Allocation :-
    Static variables and methods are allocated memory space only once during the execution of the program. This memory space is shared among all instances of the class, which makes Static members useful for maintaining global state or shared functionality.
  - Accessible without object instantiation :
    Static members can be accessed without need to create an instance of the class. This makes them useful for providing utility functions and constants that can be used across the entire program.
  - Static methods and variables :
  - Cannot access non-Static members of a class as they are not associated with any particular instance of the class

- Associated with class, not objects. This means that changes to a static member are reflected in all instanies of the class, and that you can access static member using the class name rather than an object reference

- Static methods can be overloaded, which means that you can define multiple methods with the same name but different parameters.

Q-2 can static methods be overloaded and overriden in Java? How static variables shared across multiple instances of a class?

Static method overloading:
Static method overloading occurs when you have multiple static methods in the same class with the same name but different parameters (different types or number of parameters). The method to be called is determined at compile time based on the method signature

Static method overriding:
Static methods can not be overriden in the same way instance methods can because static methods are bound to the class, not the instance. If a subclass defines a static method with the same name & parameters as a static method in a superclass, it is considered hiding, not overriding.

Static Variable :
Static variables are shared across all instances of a class. There is only one copy of the static variable, regardless of how many instances of the class are created. When one instance changes the value of a static variable, that change is reflected across all instances

~~static method overloading~~ :

Q.3 what is the significance of the final keyword in Java ?

The final keyword in Java is a modifier that can be applied to variables, methods and classes

- final variables :
when a variable is declared as final, it means that its value cannot be changed once it has been assigned. This makes the variable a constant.
- final methods :
A method declared as final cannot be overriden by subclasses
- final classes :
A class declared as final cannot be subclassed.

Q.4 what is narrowing and widening conversion in Java ?

- widening conversion :
widening conversion occurs when a smaller data type is converted to a larger data type. This type of conversion is safe because it does not result in data loss

Ex of widening conversion :
- byte to short
- short to int
- int to long
- float to double

- Narrowing Conversion :
Narrowing conversion happens when a larger data type is converted to a smaller data type. This can lead to data loss or overflow, so it must be done explicitly using a cast.
Ex of Narrowing conversion :
- int to short
- long to int
- float to byte
- double to float

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

```
public class WideningExample {
    public static void main (String [] args) {
    byte b = 10;
    int i = b;
    long l = i;
    float f = l;
    double d = f;
System.out.println ("Byte :" +b);
System.out.println ("Int :" +i);
System.out.println ("Long" +l);
System.out.println ("Float" +f);
System.out.println ("double" +d);
}
```

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

ans In Java, when performing narrowing conversions, the language requires explicit casting to indicate that you are aware of the potential loss of precision or data.

- Explicit casting:
when you convert from a large data type to a smaller one, you must use an explicit cast. This casting syntax makes it clear to the developer that there is a risk of data loss

Q.7 Explain the concept of automatic widening Conversion in Java

ans automatic widening Conversion in java refers to the process where the Java compiler automatically converts a smaller data type into a larger data type without any explicity instruction from the programmer. This conversion occurs when assigning values between compatiable primitive data types or during expression

Q.8 what are the implications of narrowing and widening conversions on type compatibility and data loss?

- widening conversions:
  - Type compatibility
    Automatic conversion: widening convertion occur automatically and dot require explicit carting. This simplifies code and enhances readability.

- Safe Compatibility : widening is safe because a large data type can accomodate all possible values of a smaller data type without risk of overflow

* Data loss
No Data loss : widening conversions do not lead to data loss or precision loss.

- Narrowing conversion :
* Type compatibility:
Explicity Casting Required : Narrowing conversions requires explicit casting, which informs the compiler (and the developer) that the conversion might lead to data lor

type safty : since narrowing can lead to lor of information it enforces a stricter approach to type safety. Developerr must ensure that the value being converted fit within the limits of the target type

* Data lor :
- potential Data lor : Narrowing conversion can lead to unexpected behaviour if the value exceedr the range of the target type. This could rerult in overflow or lors of precision.