M T W T F S S

Page No.:

Date:

YOUVA

Assignment No. 6

1) what is method overloading in Java & explain with an example?

→

(i) Method overloading is multiple methods in the same class with the same name but different Parameters.

(ii) Method overloading increases code's ~~readeo~~ readability and reusability

(iii) Inorder to overload the method we have to give different Parameter or diffierent data types of Parameter or change the Sequence of the parameters to overload the method.

(2) What are the rules for method overloading resolution in Java? How does Java determine which overloaded method to call?

→

Rules for Method overloading Resolution in Java :-

(i) To overload a method we have to consider & give parameters in this manner :-
→ Number of parameters
→ Data type of pararameter.
→ Sequence of data type of parameter

(ii) Compile time Resolution. →
The compiler decides which overloaded method should it make based on ~~Paramete~~ arguments provided by user.

Which overloaded method to call :-

(I) match between the method call & the method definition If an exact match is found. that method is called

(II) If the method not found the JVM chooses method by upcasting It never downcast the method.

(III) Method call is based on the method signature considering the number & types of parameters provided in the method call.

Q3.) What does the static keyword mean in Java? Explain the different between static & non-static methods.

→

A) Static keyword Java :-

(1) static keyword is a non-access modifier used for methods, variables, blocks & nested classes. It indicates that a particular member belongs to the class itself rather than to instances of the class. ~~[illegible]~~

(2) Static variables and methods are allocated memory space only once during program execution. This memory space is shared among all instances of the class making static members useful for maintaining global state or shared functionality.

M T W T F S S

Page No.:

Date:

YOUVA

(iii) static members can be accessed without the 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.

(iv) Static members are associated with the class, not with individual objects. Changes to a static member are reflected in all instances of the class, and static members can be accessed using the class name rather than an object refrence.

(v) static methods & variables cannot access non-static members of a class, as they are not associated with any particular instance of the class. static methods can be overloaded but not overidden because they are associated with the class rather than with a specific instance.

B) Difference betn Static & Non-static.

① Accessibility :-

→ Static methods can be accessed ~~during~~ using class names, while non-static methods require object instances ~~for~~ access.

② Memory Allocation :-

→ Static methods are allocated memory only when the class is loaded, while non-static methods have memory allocated per object instance.

(III) Sharing:-
→ All instance share static variable whereas each object has it's copy of non-static variables.

(IV) ~~So~~ scope :-
→ Static variables have global scope, while non-static variables have local scope.

(V) Method Binding:-
→ Static methods support compile-time binding, while non-static methods support dynamic or run time binding

(VI) overriding :-
→ static method cannot be overriden, but non-static methods can be overidden.

Q5.) Can static methods be overloaded and overridden in Java? How are static variables shared ~~acces~~ across multiple instances of a class?

→

(A) Static Methods overloading & overriding:-

→ overloading:- Yes static methods can be overloaded in Java. Method overloading allows defining multiple methods with the same name but different parameters within the same class.

→ overriding :→
No, static methods can not be overridden in Java. Static methods are associated with the class itself rather than with instances so they are not subject to polymorphic behaviour like instence methods.

B) Sharing of static variables :-

① Shared Memory Allocation →
Static variables in Java are allocated. memory space only once during program execution.

② Global state :→
All Instances of a class share the same static variable, making it useful for maintaining global state or shared. functionality.

③ Isolation :→
Each Instance of a class within a different class loader is distinct, meaning that static variables are shared among Instances within the same class loader but Isolated across different class loader.

Q6> what is the significance of the final keyword in Java?

→

(i) when a variable is declared as 'final' it's value cannot be changed once initialized making it's a Constant.

(ii) ~~find~~ final variables are commonly used to declare constants that should not be modified.

(iii) The naming Convention for final variables in Java is to use uppercase letters with underscores to separate words.

(iv) A method declared as final cannot be overridden by a subclass.

(v) final methods are useful for defining methods that are part of a class's public API and should not be modified by subclasses.

(a) final classes :-

→ A classes marked as 'final' cannot be inherited by another class:-

→ final classes are benficial when a classes is intended to be used as is without modification

(b) Initialization →

→ final variables must be initialized either at the time of declaration or in the constructor of the ~~classes~~. This ensures that the value of the final variable is set and cannot be changed.

(c) Performance & Security →

→ The use of 'final' can sometimes improve performance as the compiler can optimize the code more effectively when variables or methods are marked as final. 'final' can enhance security by preventing malicious code from modifying sensitive data or behaviour.

(d) Code Quality:-

→ By declaring variables, methods & classes as 'final', developers can write more secure, robust, & maintainable code

→ It ensures that certain aspects of a program remain unchanged, promoting stability & ~~rea~~ reliability.

Page No.
Date
YOUVA

Q7) Can a final method be overridden in a subclass? How does the final keyword affect variables, methods, & classes in Java?
→

A) A 'final' method cannot be overridden in a subclass. The 'final' keyword applied to a method prevents it from being overridden by any subclass, ensuring that the method's implementation remains unchanged.

B) Impact of final keyword :-

(1) Variables :-
→ final variables, once initialized, cannot be reassigned a new value.
→ Final variables are constants and are typically declared in uppercase with underscores to seperate words.
→ Bank final variables must be initialized in the constructor to avoid compilation errors.

(2) Methods :-
→ final methods cannot be overridden by subclasses, maintaining the method's behaviour across inheritance.
→ final methods are useful for defining methods that should not be modified or extended in subclasses.

Page No: Date:

③ Classes :-

→ Final classes cannot be extended by other classes, preventing inheritance

→ Final classes are beneficial when a class is designed to be used as is without modification or extension.

Q8) what does this keyword represent in Java? How is ~~the~~ this keyword used in constructors & methods?

→

ⓐ This keyword in Java represents the current object or instance of a class.

ⓑ Constructor :- 'this' is used in constructors to refer to the current object & can be used to call another constructor in the same class, facilitating constructor chaining & reducing duplicated code

ⓒ Methods :- In 'this' is used to access or modify fields of the current object especially when field names are the same as local variable names. It can also be used to pass the current object as an argument to another method or return the current object from method.

Q9> what are narrowing and widening conversions in Java?

→.

A) Widening Conversion:-

→ widening conversion changes a value to a data type that can accomodate any possible value of the original data.

→ widening conversions preserve the source value but may alter it's representation

→ Converting from an integral type to Decimal or from char to string are some examples of widening conversions.

B) Narrowing Conversions:-

→ Narrowing conversion changes a value to a data type that may not be able to hold all possible values.

→ fractional values are rounded when converted to integral types in narrowing conversions.

→ Narrowing conversions can fail at runtime or incur data loss if the destination data type cannot accommodate the converted value.

→ Narrowing conversions, when they fail, can throw exceptions like invalid cast Exception or overflowException.

Q10) provide examples of narrowing & widening Conversion between primitive data types.

→

1) widening conversions:-

Example 1 → Converting from. int to double without first converting to ~~to~~ 'long' or 'float'

eg 2 → Converting from short to int. which is a widening conversion

eg 3 → Conversion from 'char' to 'string', where the 'char' value is widened to fit into the 'string'

2) Narrowing Conversion:-

eg 1 → Conversion from a fractional type like 'double' to an Integral type like 'int' which may result in data loss

eg2 → Converting from a broader data type to a narrower data type such as conversion betn 'char', 'byte' & 'short'

eg 3 → Conversion betn Boolean & any numeric type where the numeric value is reduced to either True to false.

Q11) How does Java hadle potential loss of precision during narrowing Conversions?
→

1) Narrowing Conversions :-

→ Narrow conversions are transformations from broader ~~datt~~ data types to narrower data types, which can result in loss of information about the overall magnitude of numeric value.

→ During narrowing Conversions, Java may lose precision range, or both, depending on the data types involved

→ for example, when converting from double to float Java follows the rounding rules which can lead to precision loss & range issues

→ Java ensures that even though precision loss may occur during narrowing Conversions, they do not result in runtime exceptions.

Q 12) Explain the concept of automatic widening conversion of Java.

→

1. Automatic widening conversion occurs when two data types are automatically converted, assuring compatibility betn different data types.

2. widening conversion is implicit in Java meaning it happeans automatically without the need for explicit casting.

3. eg. converting an 'int' to a 'long' or 'float' to 'double' are common examples of widening conversions

4. when assigning an integer constant to a long variable or a floating ~~ipt~~ point constant to a float or double variable, widening comersion takes place.

5. widening conversions ensure that data can be smoothly promoted to larger data types without losing precision. ~~on~~ or range.

6. Java automatically promotes each ~~typ~~ byte, short, or char operand to ind when evaluating an expression.

7. If one operand is an expression is long, float, or double the entire expression is promoted to the corrosponding larger data type.

Q13) what are the Implications of narrowing & widening Conversions on type compatibility & data loss?

→

(A) widening Conversions :-

i) Type compatibility :-
widening conversions ensure compatibility betn different data types by changing a value to a data type that can accomodate any possible value of the original data.

(ii) Data loss :- widening Conversions preserve the source value but can alter it's representation especially when converting from an integral type to Decimal or from char to string. while they do not incur information loss, Precision may be affected in Certain Cases.

(B) Narrowing Conversions :-

i) Type compatibility :- Narrowing Conversions may lead to compatibility issues as they change a value to a data type that might not be able to hold all possible values.

ii) Data Loss :-

Narrowing conversions can result in data loss or errors if the destination data type cannot recieve the converted value. for example, a numeric conversion can lead to overflow or rounding issues, potentially causing loss of precision or range.