Java Questions & Answers – Overloading Methods & Argument Passing

1. What is process of defining two or more methods within same class that have same name but different parameters declaration?  
a) method overloading  
b) method overriding  
c) method hiding  
d) None of the mentioned  
View Answer

Answer: a  
Explanation: Two or more methods can have same name as long as their parameters declaration is different, the methods are said to be overloaded and process is called method overloading. Method overloading is a way by which Java implements polymorphism.

2. Which of these can be overloaded?  
a) Methods  
b) Constructors  
c) All of the mentioned  
d) None of the mentioned  
View Answer

Answer: c  
Explanation: None.

3. Which of these is correct about passing an argument by call-by-value process?  
a) Copy of argument is made into the formal parameter of the subroutine.  
b) Reference to original argument is passed to formal parameter of the subroutine.  
c) Copy of argument is made into the formal parameter of the subroutine and changes made on parameters of subroutine have effect on original argument.  
d) Reference to original argument is passed to formal parameter of the subroutine and changes made on parameters of subroutine have effect on original argument.  
View Answer

Answer: a  
Explanation: When we pass an argument by call-by-value a copy of argument is made into the formal parameter of the subroutine and changes made on parameters of subroutine have no effect on original argument, they remain the same.

4. What is the process of defining a method in terms of itself, that is a method that calls itself?  
a) Polymorphism  
b) Abstraction  
c) Encapsulation  
d) Recursion  
View Answer

Answer: d  
Explanation: None.

5. What is the output of the following code?

1. **class** San
2. {
3. **public** **void** m1 (**int** i,**float** f)
4. {
5. System.out.println(" int float method");
6. }
8. **public** **void** m1(**float** f,**int** i);
9. {
10. System.out.println("float int method");
11. }
13. **public** **static** **void** main(String[]args)
14. {
15. San s=**new** San();
16. s.m1(20,20);
17. }
18. }

a) int float method  
b) float int method  
c) compile time error  
d) run time error  
View Answer

Answer: c  
Explanation: While resolving overloaded method, compiler automatically promotes if exact match is not found. But in this case, which one to promote is an ambiguity.

6. What is the output of this program?

1. **class** overload {
2. **int** x;
3. **int** y;
4. **void** add(**int** a) {
5. x = a + 1;
6. }
7. **void** add(**int** a, **int** b){
8. x = a + 2;
9. }
10. }
11. **class** Overload\_methods {
12. **public** **static** **void** main(String args[])
13. {
14. overload obj = **new** overload();
15. **int** a = 0;
16. obj.add(6);
17. System.out.println(obj.x);
18. }
19. }

a) 5  
b) 6  
c) 7  
d) 8  
View Answer

Answer: c  
Explanation: None.  
output:  
$ javac Overload\_methods.java  
$ java Overload\_methods  
7

7. What is the output of this program?

1. **class** overload {
2. **int** x;
3. **int** y;
4. **void** add(**int** a){
5. x = a + 1;
6. }
7. **void** add(**int** a , **int** b){
8. x = a + 2;
9. }
10. }
11. **class** Overload\_methods {
12. **public** **static** **void** main(String args[])
13. {
14. overload obj = **new** overload();
15. **int** a = 0;
16. obj.add(6, 7);
17. System.out.println(obj.x);
18. }
19. }

a) 6  
b) 7  
c) 8  
d) 9  
View Answer

Answer: c  
Explanation: None.  
output:  
$ javac Overload\_methods.java  
$ java Overload\_methods  
8

8. What is the output of this program?

1. **class** overload {
2. **int** x;
3. **double** y;
4. **void** add(**int** a , **int** b) {
5. x = a + b;
6. }
7. **void** add(**double** c , **double** d){
8. y = c + d;
9. }
10. overload() {
11. **this**.x = 0;
12. **this**.y = 0;
13. }
14. }
15. **class** Overload\_methods {
16. **public** **static** **void** main(String args[])
17. {
18. overload obj = **new** overload();
19. **int** a = 2;
20. **double** b = 3.2;
21. obj.add(a, a);
22. obj.add(b, b);
23. System.out.println(obj.x + " " + obj.y);
24. }
25. }

a) 6 6  
b) 6.4 6.4  
c) 6.4 6  
d) 4 6.4  
View Answer

Answer: d  
Explanation: For obj.add(a,a); ,the function in line number 4 gets executed and value of x is 4. For the next function call, the function in line number 7 gets executed and value of y is 6.4  
output:  
$ javac Overload\_methods.java  
$ java Overload\_methods  
4 6.4

9. What is the output of this program?

1. **class** test {
2. **int** a;
3. **int** b;
4. **void** meth(**int** i , **int** j) {
5. i \*= 2;
6. j /= 2;
7. }
8. }
9. **class** Output {
10. **public** **static** **void** main(String args[])
11. {
12. test obj = **new** test();
13. **int** a = 10;
14. **int** b = 20;
15. obj.meth(a , b);
16. System.out.println(a + " " + b);
17. }
18. }

a) 10 20  
b) 20 10  
c) 20 40  
d) 40 20  
View Answer

Answer: a  
Explanation: Variables a & b are passed by value, copy of their values are made on formal parameters of function meth() that is i & j. Therefore changes done on i & j are not reflected back on original arguments. a & b remain 10 & 20 respectively.  
output:  
$ javac Output.java  
$ java Output  
10 20

10. What is the output of this program?

1. **class** test {
2. **int** a;
3. **int** b;
4. test(**int** i, **int** j) {
5. a = i;
6. b = j;
7. }
8. **void** meth(test o) {
9. o.a \*= 2;
10. O.b /= 2;
11. }
12. }
13. **class** Output {
14. **public** **static** **void** main(String args[])
15. {
16. test obj = **new** test(10 , 20);
17. obj.meth(obj);
18. System.out.println(obj.a + " " + obj.b);
19. }
20. }

a) 10 20  
b) 20 10  
c) 20 40  
d) 40 20  
View Answer

Answer: b  
Explanation: class objects are always passed by reference, therefore changes done are reflected back on original arguments. obj.meth(obj) sends object obj as parameter whose variables a & b are multiplied and divided by 2 respectively by meth() function of class test. a & b becomes 20 & 10 respectively.  
output:  
$ javac Output.java  
$ java Output  
20 10