**Constructors**

**Points to remember**

* If you define no constructors at all, then the compiler provides a default constructor with no arguments. Even if, you define one constructor, this default is not provided.
* A constructor can call other overloaded constructors by ‘**this (arguments)**’. If you use this, it must be the first statement in the constructor. This construct can be used only from within a constructor.
* A constructor can’t call the same constructor from within. Compiler will say ‘ **recursive constructor invocation**’
* Constructors can’t have a return type. A method with a class name, but with a return type is not considered a constructor, but just a method by compiler. Expect trick questions using this.
* **Constructor body can have an empty return statement**. Though void cannot be specified with the constructor signature, empty return statement is acceptable.
* Only modifiers that a constructor can have are the accessibility modifiers.
* Constructors cannot be overridden, since they are not inherited.

**Example 2.3.1E (M) Level: 1**

Purpose

How to define a constructor, and used for initializing the instance variables.

Code

1. **class** BulbUser {
2. *// instance variables*
3. **int** watts;
4. **private** **boolean** status = false;
5. *// constructor defined*
6. BulbUser() {
7. System.out.println("A bulb has been created");
8. }
9. *// constructor with a parameter*
10. BulbUser(int watts) {
11. **this**.watts = watts; *//variable referred by this is instance*

// *variable of class*

1. }
2. *// instance methods*
3. **void** switchOn() {
4. status = true;
5. }
6. **void** switchOff() {
7. status = false;
8. }
9. **void** bulbDetails() {
10. System.out.println("Bulb Wattage: " + watts);
11. **if**(status)
12. System.out.println("Bulb is On.");
13. **else**
14. System.out.println("Bulb is Off.");
15. }
16. }
17. **class** Main {
18. **public** **static** **void** main(String args[]){
19. BulbUser bulb1 = **new** BulbUser(); *// constructor will be called*
20. BulbUser bulb1 = **new** BulbUser(60); *//invokes the constructor*
21. bulb1.bulbDetails();
22. }
23. }

Output

A bulb has been created

Bulb Wattage: 60

Bulb is Off.

Check Your Understanding

1. In the above example a class named BulbUser is defined with instance variables watts and status. The variable status can be accessed within class BulbUser only because it is private thus providing data hiding.
2. A constructor is also defined with a simple output statement.(see line no’s 6 to 8)
3. The name of the constructor is same as the name of …………..(see line 6)
4. Identify the two classes and variables declared in this example.
5. A constructor is also defined which takes ……… parameter and assigns the value to the instance variable …………... (see line no. 6 to 8).
6. An object of class BulbUser is defined in class ………… (see line no 24) and value 60 is passed as argument. (see line no 26)

 Check: Do you now understand

* + How to define a constructor.
  + When the constructor is invoked.
  + How to define a constructor with parameter.
  + Use of **this** keyword.
  + Constructors mainly serve the purpose of initialization of object.

**Example 2.3.2E (M) Level: 2**

Purpose

Generating an erroneous program showing that default constructor does not exist in every case.

Code

1. **class** BulbUser {
2. *// instance variables*
3. **int** watts;
4. **private** **boolean** status = false;
5. *// constructor with a parameter*
6. BulbUser(**int** watts) {
7. this.watts = watts;
8. }
9. *// instance methods*
10. **void** switchOn() {
11. status = true;
12. }
13. **void** switchOff() {
14. status = false;
15. }
16. **void** bulbDetails() {
17. System.out.println("Bulb Wattage: " + watts);
18. **if**(status)
19. System.out.println("Bulb is On.");
20. **else**
21. System.out.println("Bulb is Off.");
22. }
23. }
24. **class** Main {
25. **public** **static** **void** main(String args[]){
26. BulbUser bulb1 = **new** BulbUser();
27. *// attempts to call a constructor without a parameter which is not defined in // the class BulbUser*
28. bulb1.watts = 100;
29. bulb1.bulbDetails();
30. }
31. }

Output

Main.java:27: cannot resolve symbol

symbol : constructor BulbUser ()

location: class BulbUser

BulbUser bulb1 = new BulbUser();

^

1 error

Check Your Understanding

1. Now, can you create a class, declare variables and initialize them.
2. Can you now define the constructor of the class and initialize class variables.
3. An object of class BulbUser is defined in class ………… (see line no 24) without any argument. This will invoke a call to a constructor without a parameter which is not defined. (see line no 26)

 Check: Do you now understand

* + Default constructor is not provided by JVM in all cases.

**Example 2.3.3E (M) Level: 3**

Purpose

Overloading the constructor method in multiple versions.

Code

1. *// This method can be overloaded to accommodate first and last names or only the last name:*
2. **class** Person {
3. private String firstName;
4. private String lastName;
5. *// Constructor for initializing first and last names*.
6. Person() {
7. **this**.firstName = "";
8. **this**.lastName = "";
9. }
10. *//Overload method Person for last name parameter*.
11. Person(String lname)
12. {
13. **this**.firstName = "";
14. **this**.lastName = lname;
15. }
16. *// Overload method Person for first and last name parameter*.
17. Person (String fname, String lname) {
18. **this**.firstName = fname;
19. **this**.lastName = lname;
20. }
21. }
22. **class** Overload {
23. **public static void** main(String args[]){
24. Person p1 = **new** Person();
25. Person p2 = **new** Person("Surya");
26. Person p3 = **new** Person("Surya", "Prakash");
27. }
28. }

Output

No Output for the above program.

// write print statement to see the output

Check Your Understanding

1. A constructor is defined which takes no parameter and assigns the value of the instance variables to null (see line no. 6 to 9).
2. A constructor is defined which takes one parameter and assigns the value of lastName with the value passed (see line no. 17 to 21).
3. A constructor is defined which takes two parameter and assigns the value of the instance variables with the values passed (see line no. 6 to 9).
4. Construtors having same name but differ in the number and/or order of parameters demonstrate the concept of ……………………..

 Check: Do you now understand

* + How constructors can be overloaded.
  + How different constructors will be invoked.

**Example 2.3.4E (M) Level: 3**

Purpose

Invoking a constructor using **this** keyword (Constructor redirecting).

Code

1. **class** BulbUser {
2. *// instance variables*
3. **int** watts;
4. **private** **boolean** status = false;
5. *// constructor with no parameter*
6. BulbUser() {
7. **this**(60);
8. System.out.println("Constructor with no paramater");
9. }
10. *// constructor with a parameter*
11. BulbUser(**int** watts) {
12. **this**.watts = watts;
13. System.out.println("Constructor with one paramater");
14. }
15. *// instance methods*
16. **void** switchOn() {
17. status = true;
18. }
19. **void** switchOff() {
20. status = false;
21. }
22. **void** bulbDetails() {
23. System.out.println("Bulb Wattage: " + watts);
24. **if**(status)
25. System.out.println("Bulb is On.");
26. **else**
27. System.out.println("Bulb is Off.");
28. }
29. }
30. **class** Main {
31. **public** **static** **void** main(String args[]){
32. BulbUser bulb1 = **new** BulbUser();
33. *// attempts to call a constructor without a parameter*
34. bulb1.watts = 100;
35. bulb1.bulbDetails();
36. }
37. }

Output

Constructor with one paramater

Constructor with no paramater

Bulb Wattage: 100

Bulb is Off.

Check Your Understanding

1. How many constructors do we have here? ( see line no. 6, 11)
2. First constructor takes no parameter and the other takes ………………

(see line no. 11). It initializes a class variable using keyword ……………….

(see line no. 12)

1. The class is instantiated in class Main with constructor having no parameters. Can you identify how the variable is initialized and method is invoked. (see line no. 34, 35)

 Check: Do you now understand

* + How to invoke a constructor using this keyword.
  + Concept of constructor redirecting.

**Quiz**

**1. What is the output of the following code?**

1. class Test{
2. Test(int i) {
3. System.out.println("Test(" +i +")");
4. }
5. }
6. public class Q12{
7. static Test t1 = new Test(1);
8. Test t2 = new Test(2);
9. static Test t3 = new Test(3);
10. public static void main(String[] args){
11. Q12 Q = new Q12();
12. }
13. }
    1. Test(1)

Test(2)

Test(3)

* 1. Test(3)

Test(2)

Test(1)

* 1. Test(2)

Test(1)

Test(3)

* 1. Test(1)

Test(3)

Test(2)

**2. What will happen if you compile/run the following code?**

1. public class Q21{
2. int maxElements;
3. void Q21(){
4. maxElements = 100;
5. System.out.println(maxElements);
6. }
7. Q21(int i){
8. maxElements = i;
9. System.out.println(maxElements);
10. }
11. public static void main(String[] args){
12. Q21 a = new Q21();
13. Q21 b = new Q21(999);
14. }
15. }
    1. Prints 100 and 999.
    2. Prints 999 and 100.
    3. Compilation error at line 3, variable maxElements was not initialized.
    4. Compillation error at line 19.

**3. Which three statements are true? (Choose three.)**

A. The default constructor initialises method variables.

B. The default constructor has the same access as its class.

C. The default constructor invokes the no-arg constructor of the superclass.

D. If a class lacks a no-arg constructor, the compiler always creates a default constructor.

E. The compiler creates a default constructor only when there are no other constructors for the class.

**4. Given:**

01. public class Test { }

**What is the prototype of the default constructor?**

A. Test( )

B. Test(void)

C. public Test( )

D. public Test(void)

E. public void Test( )

**5. Given:**

01. public class A{

02. void A() {

03. System.out.println("Class A");

04. }

05. public static void main(String[] args) {

06. new A();

07. }

08. }

**What is the result?**

A. Class A

B. Compilation fails.

C. An exception is thrown at line 2.

D. An exception is thrown at line 6.

E. The code executes with no output

**6. this ( ) method is used to call a constructor from the constructor of same class**

1. True
2. False

**7. Under what situations do you obtain a default constructor?**

1. When you define any class
2. When the class has no other constructors
3. When you define at least one constructor

**8. Given the following code:**

1. public class Test {
2. …
3. }

**Which of the following can be used to define a constructor for this class:**

1. public void Test() {…}
2. public Test() {…}
3. public static Test() {…}
4. public static void Test() {…}

**9. Where in a constructor, can you place a call to a constructor defined in the super class?**

1. Anywhere
2. The first statement in the constructor
3. The last statement in the constructor
4. You can't call super in a constructor

**Answers**

1. D

Explanation : No matter where they declared, static variables will be intitialized before non-static variables.

2. D

Explanation : Constructors should not return any value. Java won't allow to indicate with void. In this case void Q21() is an ordinary method which has the same name of the Class.

3. B, C, E

Explanation: B sounds correct as in the example below class CoffeeCup { private int innerCoffee; public CoffeeCup() { } public void add(int amount) { innerCoffee += amount; } //... } The compiler gives default constructors the same access level as their class. In the example above, class CoffeeCup is public, so the default constructor is public. If CoffeeCup had been given package access, the default constructor would be given package access as well. C is correct. The Java compiler generates at least one instance initialisation method for every class it compiles. In the Java class file, the instance initialisation method is named "<init>." For each constructor in the source code of a class, the Java compiler generates one <init>() method. If the class declares no constructors explicitly, the compiler generates a default no-arg constructor that just invokes the superclass's no-arg constructor. As with any other constructor, the compiler creates an <init>() method in the class file that corresponds to this default constructor. E is correct. The compiler creates a default constructor if you do not declare any constructors in your class.

4. C

5. E

Explanation: E is correct. The specification at line 2 is for a method and not a constructor and this method are never called therefore there is no output. The constructor that is called is the default constructor.

6. True

7. B

8. B

9. B

**Rapid Fire**

1. Develop a program having a class without defining a constructor?
2. Develop a program having a class overloaded another class and call the overloaded constructor?
3. Develop a program having a class and constructor call the same constructor from within same?
4. Can you try to override a constructor?

**References**

Java Complete Reference 🡪 Chapter 6 🡪 Page No: 145

Java Complete Reference 🡪 Chapter 7 🡪 Page No: 159