Object and Objects Classes

Exercises and Solutions

1. What is the fully qualified name of the class that is the superclass of all classes in Java?

**Answer:**

java.lang.Object

1. What is the superclass of the java.lang.Object class?

**Answer:**

The java.lang.Object class does not have a superclass.

1. Name three methods that are available in the Object class and describe their usage in brief.

**Answer:**

* notifyAll() – Notifies all threads in the wait queue of the object.
* toString() – Returns a string representation of an object.
* equals() – Compare two objects for equality.

1. What is a hash code? When is it used in Java? What method in the Object class is used to return the hash code of an object?

**Answer:**

A hash code is an integer value that is computed for a piece of information using an algorithm. Java uses hash codes to efficiently retrieve data from hash based collections. The hashCode() method in the Object class returns hash code of an object.

1. How is the comparison of two objects performed using the == operator?

**Answer:**

The equality operator, ==, always compares the references of its two operands.

1. What method of the Object class must be overridden in your class if you want to compare objects of your class for equality based on their state, not their references?

**Answer:**

The equals() method

1. What is the default implementation of the equals() method in the Object class?

**Answer:**

The default implementation of the equals() method compares the references of the object being passed as the parameter and the object on which the method is called.

public boolean equals(Object obj) {

return (this == obj);

}

1. Is the following statement true in Java?  
     
   If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.

**Answer:**

Yes. If x.equals(y) returns true, x.hashCode() must return the same value as y.hashCode().

1. If your class overrides the equals() method of the Object class, which other method of the Object class should also be overridden by your class?

**Answer:**

The hashCode() method.

1. What is the cloning of objects in Java? What are shallow and deep cloning?

**Answer:**

Cloning an object means copying the content of the object bit by bit. When an object is cloned without cloning the contents of the objects the first object references, it is called shallow closing. When the contents of an objects and the contents of all other objects that the first object references are cloned, it is called deep cloning.

1. What method of the Object class you need to override in your class to allow cloning of objects of your class? Create a Phone class with two fields as shown:  
     
   // Phone.java  
   package com.jdojo.object.excercise;  
     
   public class Phone {  
    private String areaCode;  
    private String number;  
   }  
     
   Implement the clone() method in the Phone class, so the Phone objects can be cloned correctly. Both instance variables in the class are required.

**Solution:**

The Phone class needs to override the clone() method of the Object class. The Phone class should also implement the Cloneable interface. The following code for the Phone class also overrides the toString() method and contains a main() method to test cloning of a Phone.

// Phone.java

package com.jdojo.object.excercise;

public class Phone implements Cloneable{

private String areaCode;

private String number;

public Phone() {

}

public Phone(String areaCode, String number) {

this.areaCode = areaCode;

this.number = number;

}

public static void main(String[] args) {

Phone p = new Phone("334", "4569067");

Phone clonedPhone = p.clone();

System.out.println("Phone: " + p);

System.out.println("Clonsed Phone: " + clonedPhone);

}

@Override

public Phone clone() {

Phone copy = null;

try {

copy = (Phone) super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

return copy;

}

@Override

public String toString() {

return this.areaCode + this.number;

}

}

1. What method of the Object class you need to override to provide a string representation of objects of your class? Enhance the Phone class by implementing the toString() method.

**Answer:**

You need to override the toString() method of the Object class to provide a string representation of the objects of your class. Refer to the solution of the previous exercise for the implementation of the toString() method of the Phone class.

1. What is the use of the finalize() method in a class? Should you use the finalize() method to cleanup resources held by objects of your class?

**Answer:**

The garbage collector calls the finalize() method before an object is destroyed. No. The finalize() method has been deprecated since Java 9 and you should bot use it to clean up resources anymore. Use try-with-resources and try-finally blocks to perform the cleanup logic in your class.

1. What is an immutable object and an immutable class? What are the benefits of using immutable objects? Name one immutable class in Java that you use very often.

**Answer:**

An object whose state cannot be changed after it is created is called an immutable object. A class whose objects are immutable is called an immutable class.

Immutable objects can be shared by different areas of a program without worrying about its state changes. Immutable object is thread-safe. Testing an immutable class is easy.

The java.lang.String class is an immutable class.

1. Use the methods of the Objects class to implement the hashCode() method and other methods of the Phone class. For example, use the requireNonNull() method of the Objects class inside constructors and methods of the Phone class to validate arguments' values.

**Solution:**

// Phone.java

package com.jdojo.object.excercise;

import java.util.Objects;

public class Phone implements Cloneable {

private String areaCode;

private String number;

public Phone() {

}

public Phone(String areaCode, String number) {

**Objects.requireNonNull(areaCode, "Area code is required.");**

**Objects.requireNonNull(number, "Number is required.");**

this.areaCode = areaCode;

this.number = number;

}

public static void main(String[] args) {

Phone p = new Phone("334", "4569067");

Phone clonedPhone = p.clone();

System.out.println("Phone: " + p);

System.out.println("Clonsed Phone: " + clonedPhone);

}

@Override

public Phone clone() {

Phone copy = null;

try {

copy = (Phone) super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

return copy;

}

**@Override**

**public int hashCode() {**

**return Objects.hash(areaCode, number);**

**}**

**@Override**

**public boolean equals(Object other) {**

**return Objects.equals(this, other);**

**}**

@Override

public String toString() {

return this.areaCode + this.number;

}

}

1. Write the missing pieces the following snippet of code that will prints the simple name and the fully qualified name of the Phone class:  
     
   Phone p = new Phone();  
   Class cls = /\* your code goes here \*/;  
   String simpleName = cls./\* your code goes here \*/;  
   String fullyQualifedName = cls./\* your code goes here \*/;  
   System.out.println("Simple class name: " + simpleName); System.out.println("Fully qualified name: " + fullyQualifedName);

**Solution:**

Phone p = new Phone();

Class cls = p.getClass();

String simpleName = cls.getSimpleName();

String fullyQualifedName = cls.getName();

System.out.println("Simple class name: " + simpleName);

System.out.println("Fully qualified name: " + fullyQualifedName);