**//UPcasting and Downcasting with instanceOf**

1. **interface** Printable{}
2. **class** A **implements** Printable{
3. **public** **void** a(){System.out.println("a method");}
4. }
5. **class** B **implements** Printable{
6. **public** **void** b(){System.out.println("b method");}
7. }
9. **class** Call{
10. **void** invoke(Printable p){//upcasting
11. **if**(p **instanceof** A){
12. A a=(A)p;//Downcasting
13. a.a();
14. }
15. **if**(p **instanceof** B){
16. B b=(B)p;//Downcasting
17. b.b();
18. }
20. }
21. }//end of Call class
23. **class** Test4{
24. **public** **static** **void** main(String args[]){
25. Printable p=**new** B();
26. Call c=**new** Call();
27. c.invoke(p);
28. }
29. }

Nested Interfaces:

https://www.javatpoint.com/nested-interface

Example of abstract class and interface in Java

Let's see a simple example where we are using interface and abstract class both.

1. //Creating interface that has 4 methods
2. **interface** A{
3. **void** a();//bydefault, public and abstract
4. **void** b();
5. **void** c();
6. **void** d();
7. }
9. //Creating abstract class that provides the implementation of one method of A interface
10. **abstract** **class** B **implements** A{
11. **public** **void** c(){System.out.println("I am C");}
12. }
14. //Creating subclass of abstract class, now we need to provide the implementation of rest of the methods
15. **class** M **extends** B{
16. **public** **void** a(){System.out.println("I am a");}
17. **public** **void** b(){System.out.println("I am b");}
18. **public** **void** d(){System.out.println("I am d");}
19. }
21. //Creating a test class that calls the methods of A interface
22. **class** Test5{
23. **public** **static** **void** main(String args[]){
24. A a=**new** M();
25. a.a();
26. a.b();
27. a.c();
28. a.d();
29. }}

### Methods of Object class

|  |
| --- |
| The Object class provides many methods. They are as follows: |

|  |  |
| --- | --- |
| **Method** | **Description** |
| public final Class getClass() | returns the Class class object of this object. The Class class can further be used to get the metadata of this class. |
| public int hashCode() | returns the hashcode number for this object. |
| public boolean equals(Object obj) | compares the given object to this object. |
| protected Object clone() throws CloneNotSupportedException | creates and returns the exact copy (clone) of this object. |
| public String toString() | returns the string representation of this object. |
| public final void notify() | wakes up single thread, waiting on this object's monitor. |
| public final void notifyAll() | wakes up all the threads, waiting on this object's monitor. |
| public final void wait(long timeout)throws InterruptedException | causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait(long timeout,int nanos)throws InterruptedException | causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait()throws InterruptedException | causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method). |
| protected void finalize()throws Throwable | is invoked by the garbage collector before object is being garbage collected. |

Example of clone() method (Object cloning)

Let's see the simple example of object cloning

1. **class** Student18 **implements** Cloneable{
2. **int** rollno;
3. String name;
5. Student18(**int** rollno,String name){
6. **this**.rollno=rollno;
7. **this**.name=name;
8. }
10. **public** Object clone()**throws** CloneNotSupportedException{
11. **return** **super**.clone();
12. }
14. **public** **static** **void** main(String args[]){
15. **try**{
16. Student18 s1=**new** Student18(101,"amit");
18. Student18 s2=(Student18)s1.clone();
20. System.out.println(s1.rollno+" "+s1.name);
21. System.out.println(s2.rollno+" "+s2.name);
23. }**catch**(CloneNotSupportedException c){}
25. }
26. }

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | [char charAt(int index)](https://www.javatpoint.com/java-string-charat) | returns char value for the particular index |
| 2 | [int length()](https://www.javatpoint.com/java-string-length) | returns string length |
| 3 | [static String format(String format, Object... args)](https://www.javatpoint.com/java-string-format) | returns formatted string |
| 4 | [static String format(Locale l, String format, Object... args)](https://www.javatpoint.com/java-string-format) | returns formatted string with given locale |
| 5 | [String substring(int beginIndex)](https://www.javatpoint.com/java-string-substring) | returns substring for given begin index |
| 6 | [String substring(int beginIndex, int endIndex)](https://www.javatpoint.com/java-string-substring) | returns substring for given begin index and end index |
| 7 | [boolean contains(CharSequence s)](https://www.javatpoint.com/java-string-contains) | returns true or false after matching the sequence of char value |
| 8 | [static String join(CharSequence delimiter, CharSequence... elements)](https://www.javatpoint.com/java-string-join) | returns a joined string |
| 9 | [static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)](https://www.javatpoint.com/java-string-join) | returns a joined string |
| 10 | [boolean equals(Object another)](https://www.javatpoint.com/java-string-equals) | checks the equality of string with object |
| 11 | [boolean isEmpty()](https://www.javatpoint.com/java-string-isempty) | checks if string is empty |
| 12 | [String concat(String str)](https://www.javatpoint.com/java-string-concat) | concatinates specified string |
| 13 | [String replace(char old, char new)](https://www.javatpoint.com/java-string-replace) | replaces all occurrences of specified char value |
| 14 | [String replace(CharSequence old, CharSequence new)](https://www.javatpoint.com/java-string-replace) | replaces all occurrences of specified CharSequence |
| 15 | [static String equalsIgnoreCase(String another)](https://www.javatpoint.com/java-string-equalsignorecase) | compares another string. It doesn't check case. |
| 16 | [String[] split(String regex)](https://www.javatpoint.com/java-string-split) | returns splitted string matching regex |
| 17 | [String[] split(String regex, int limit)](https://www.javatpoint.com/java-string-split) | returns splitted string matching regex and limit |
| 18 | [String intern()](https://www.javatpoint.com/java-string-intern) | returns interned string |
| 19 | [int indexOf(int ch)](https://www.javatpoint.com/java-string-indexof) | returns specified char value index |
| 20 | [int indexOf(int ch, int fromIndex)](https://www.javatpoint.com/java-string-indexof) | returns specified char value index starting with given index |
| 21 | [int indexOf(String substring)](https://www.javatpoint.com/java-string-indexof) | returns specified substring index |
| 22 | [int indexOf(String substring, int fromIndex)](https://www.javatpoint.com/java-string-indexof) | returns specified substring index starting with given index |
| 23 | [String toLowerCase()](https://www.javatpoint.com/java-string-tolowercase) | returns string in lowercase. |
| 24 | [String toLowerCase(Locale l)](https://www.javatpoint.com/java-string-tolowercase) | returns string in lowercase using specified locale. |
| 25 | [String toUpperCase()](https://www.javatpoint.com/java-string-touppercase) | returns string in uppercase. |
| 26 | [String toUpperCase(Locale l)](https://www.javatpoint.com/java-string-touppercase) | returns string in uppercase using specified locale. |
| 27 | [String trim()](https://www.javatpoint.com/java-string-trim) | removes beginning and ending spaces of this string. |
| 28 | [static String valueOf(int value)](https://www.javatpoint.com/java-string-valueof) | converts given type into string. It is overloaded. |

## **Performance Test of String and StringBuffer**

1. **public** **class** ConcatTest{
2. **public** **static** String concatWithString()    {
3. String t = "Java";
4. **for** (**int** i=0; i<10000; i++){
5. t = t + "Tpoint";
6. }
7. **return** t;
8. }
9. **public** **static** String concatWithStringBuffer(){
10. StringBuffer sb = **new** StringBuffer("Java");
11. **for** (**int** i=0; i<10000; i++){
12. sb.append("Tpoint");
13. }
14. **return** sb.toString();
15. }
16. **public** **static** **void** main(String[] args){
17. **long** startTime = System.currentTimeMillis();
18. concatWithString();
19. System.out.println("Time taken by Concating with String: "+(System.currentTimeMillis()-startTime)+"ms");
20. startTime = System.currentTimeMillis();
21. concatWithStringBuffer();
22. System.out.println("Time taken by Concating with  StringBuffer: "+(System.currentTimeMillis()-startTime)+"ms");
23. }
24. }

Time taken by Concating with String: 578ms

Time taken by Concating with StringBuffer: 0ms