Java Tricky Output Questions

**Question 1:** What will be the Output of the below code:

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| public class A {      public static void main(String[] args)      {          if (true)              break;      }  } |

**Choices:**

* a) Nothing
* b) Error

**Answer:** b) Error  
**Reason:** Break statement can only be used with loop or switch. So, using break with if statement causes “break outside switch or loop” error.

**Question 2:** What will be the Output of the below code:

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| public class A {      public static void main(String[] args)      {          System.out.println('j' + 'a' + 'v' + 'a');      }  } |

**Choices:**

* a) java
* b) Something else (Other than simple concatenation)

**Answer:** b) Something else (Other than simple concatenation)  
**Reason:** “java” would be printed if String literals (in double quotes) are used, but in the question since character literals has been used, these won’t be concatenated. Therefore After execution of the program, an addition of each equivalent ASCII(Unicode) value of the character will be obtained.  
Hence the output is **106 + 97 + 118 + 97 = 418**

**Question 3:** What will be the Output of the below code:

|  |
| --- |
| public class A {      public static void main(String[] args)      {          int $\_ = 5;      }  } |

**Choices:**

* a) Nothing
* b) Error

**Answer:** a) Nothing  
**Reason:** It looks like $ will cause an error, but it won’t. In java, identifier rule says, **identifier can start with any alphabet or underscore (“\_”) or dollar (“$”)**. So answer is Nothing.

**Question 4:** What will be the Output of the below code:

|  |
| --- |
| public class Demo{      public static void main(String[] arr){          Integer num1 = 100;          Integer num2 = 100;          Integer num3 = 500;          Integer num4 = 500;            if(num1==num2){              System.out.println("num1 == num2");          }          else{              System.out.println("num1 != num2");          }          if(num3 == num4){              System.out.println("num3 == num4");          }          else{              System.out.println("num3 != num4");          }      }  } |

**Choices:**

* a) num1 == num2  
      num3 == num4
* b) num1 == num2  
      num3 != num4
* c) num1 != num2  
      num3 == num4
* d) num1 != num2  
      num3 != num4

**Answer:** b)num1 == num2  
                  num3 != num4  
**Reason:** We always thought that whenever two object references are compared using “==”, it always evaluates to “false”. But here Integer caching changes the results.**Integer class has a caching range of -128 to 127.** Whenever a number is between this range and autoboxing is used, it assigns the same reference. That’s why for value 100, both num1 and num2 will have the same reference, but for the value 500 (not in the range of -128 to 127), num3 and num4 will have different reference.

**Question 5:** What will be the Output of the below code:

|  |
| --- |
| public class Demo{      public static void main(String[] arr){        }      public static void main(String arr){        }  } |

**Choices:**

* a) Nothing
* b) Error

**Answer:** a) Nothing  
**Reason:** We can overload main() too. But JVM will always call main() that has String[] argument.

ListIterator Example:

----------------------------------

import java.util.ArrayList;

import java.util.List;

import java.util.ListIterator;

public class ListIteratorExample {

public static void main(String a[]){

ListIterator<String> litr = null;

List<String> names = new ArrayList<String>();

names.add("Shyam");

names.add("Rajat");

names.add("Paul");

names.add("Tom");

names.add("Kate");

//Obtaining list iterator

litr=names.listIterator();

System.out.println("Traversing the list in forward direction:");

while(litr.hasNext()){

System.out.println(litr.next());

}

System.out.println("\nTraversing the list in backward direction:");

while(litr.hasPrevious()){

System.out.println(litr.previous());

}

}

}