L45

Friday, February 4, 2022 5:20 PM

Understanding System Class

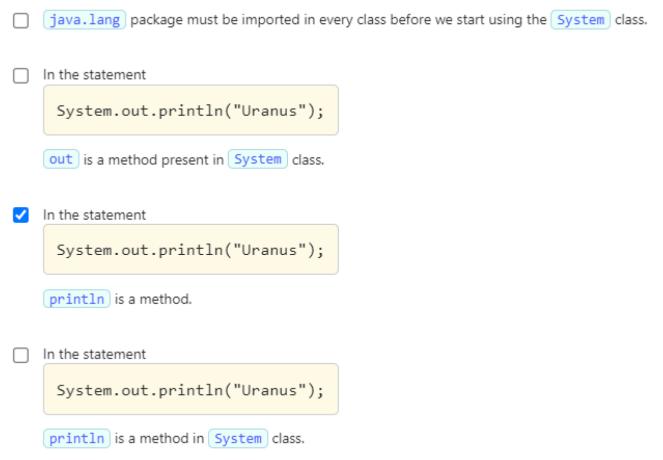
Q1. Among the various classes available in java.lang package, System is one of the most commonly used classes after String.

The System class can neither be **instantiated** (since it has a private constructor) nor can be **extended** (since it is declared as final).

System class has three important public static fields:

1. **out** - is accessed as System.out. This out field is of type PrintStream. The out refers to the standard output stream.
2. **err** - is accessed as System.err. This err field is of type PrintStream. The err refers to the standard error stream.
3. **in** - is accessed as System.in. This in field is of type InputStream. The in refers to the standard input stream.

The System class provides a method called System.console(), which returns the java.io.Console object associated with the running Java process (Java Virtual Machine). Select all the correct statement given below.



[ ] `java.lang` package must be imported in every class before we start using the `System` class.

[ ] In the statement

```
System.out.println("Uranus");
```

`out` is a method present in `System` class.

[✓] In the statement

```
System.out.println("Uranus");
```

`println` is a method.

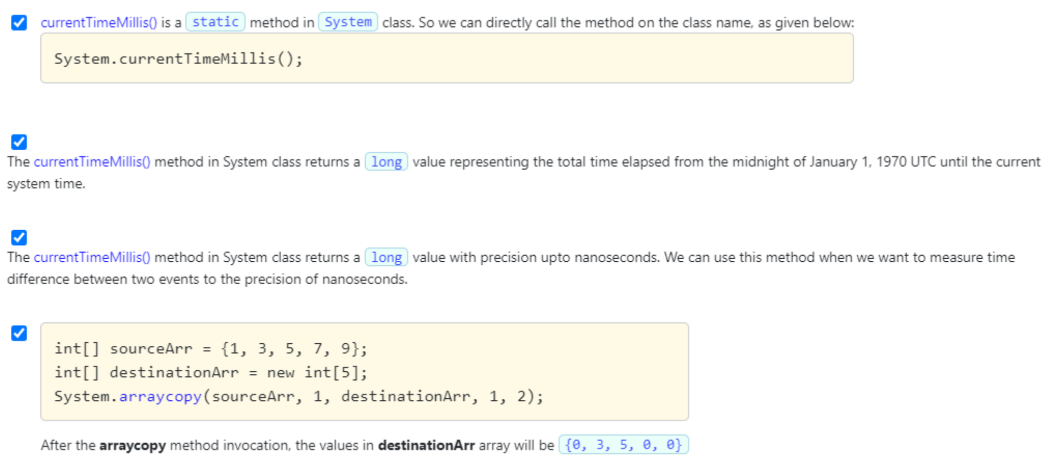
[ ] In the statement

```
System.out.println("Uranus");
```

`println` is a method in `System` class.

Q2. Common methods in System class

Select all the correct statements for some of the common methods in System class. [Hint: Make sure to click on the method names and read the method documentation before you mark the answers].



[✓] currentTimeMillis() is a `static` method in `System` class. So we can directly call the method on the class name, as given below:

```
System.currentTimeMillis();
```

[✓] The currentTimeMillis() method in System class returns a `long` value representing the total time elapsed from the midnight of January 1, 1970 UTC until the current system time.

[✓] The currentTimeMillis() method in System class returns a `long` value with precision upto nanoseconds. We can use this method when we want to measure time difference between two events to the precision of nanoseconds.

[✓]

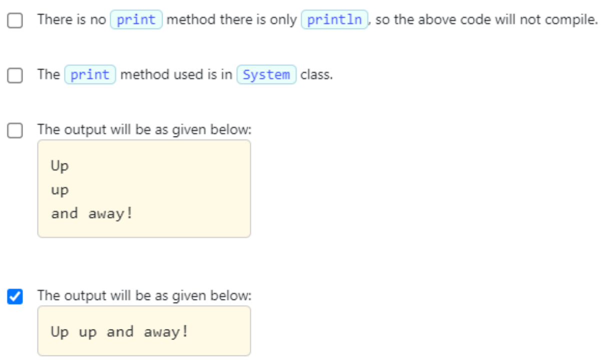
```
int[] sourceArr = {1, 3, 5, 7, 9};
int[] destinationArr = new int[5];
System.arraycopy(sourceArr, 1, destinationArr, 1, 2);
```

After the **arraycopy** method invocation, the values in **destinationArr** array will be `{0, 3, 5, 0, 0}`

Truths about println method

Q1. Select all the correct statements for the below code:

```
public class Demo {
        public static void main(String[] args) {
                System.out.print("Up ");
                System.out.print("up");
                System.out.print(" and away!");
        }
}
```



- ☐ There is no `print` method there is only `println`, so the above code will not compile.
- ☐ The `print` method used is in `System` class.
- ☐ The output will be as given below:

```
Up
up
and away!
```

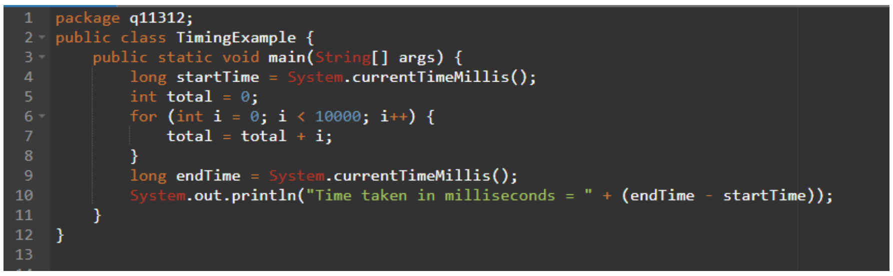
- ☑ The output will be as given below:

```
Up up and away!
```

Working with System Time in Milli and Nano seconds

Q1. In many situations we would like to measure the time taken to perform a certain operation. Below is an example which uses the methods present in the System class to measure the time taken.

See and retype the below code to understand the usage of System.currentTimeMillis() and System.nanoTime()



```java
package q11312;
public class TimingExample {
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();
        int total = 0;
        for (int i = 0; i < 10000; i++) {
            total = total + i;
        }
        long endTime = System.currentTimeMillis();
        System.out.println("Time taken in milliseconds = " + (endTime - startTime));
    }
}
```