**System Class:**

is one of the core classes. It is final and all it’s members and methods are static so that we can’t subclass and override it’s behavior through inheritance.

System class contains very useful methods which provides System information

* methods to get the list of System properties. Get specific property, set system property and clear any existing property.
* provides a method to get the environment variables data in Map form and it contains key-value pairs in String object.
* Provides a method to get unique Console object associated with the running JVM. ( If no console is associated with current JVM, for example running through IDE or running as background program, then it returns null).it is null if it runs in IDE)
* contains three fields- in, out and err. They are used to read data from InputStream and to write data to OutputStream.
* Provides two methods to get the current time in milliseconds and nano time. Method to copy partial array to another array.
* SecurityManager class is used to implement security policy for applications, System class provide useful methods to get SecurityManager for the currently running JVM and to set the SecurityManager for the application.

**To get System defined properties:**

System.***out***.println(System.*getProperties*());

Or // to get in formatted way

Properties p=System.*getProperties*();  
Set<Object> set=p.keySet();  
**for**(Object obj:set) {  
 String key =(String) obj;  
 System.***out***.println( obj + **" = "** +System.*getProperty*(key));

}

**To get specific property:**

System.***out***.println(System.*getProperty*(“user.name”));

**To clear property:**

System.clearProperty(“user.country”);

**To get unmodified environment variables :**

System.***out***.println(System.*getenv*());

Or // to get in formatted way

Map<String, String> m=System.*getenv*();  
Set<String> s=m.keySet();  
**for**(String str:s) {  
 System.***out***.println(str +**" : "** +m.get(str));  
}

**To get specific environment variable:**

System.***out***.println(System.*getenv*(“PATH”));

**To get free memory:**

Runtime r=Runtime.*getRuntime*();  
System.***out***.println(r.freeMemory());

**To get current time in miliiseconds.**

System.***out***.println(**"current time in milliseconds : "**+ System.*currentTimeMillis*());

**To get date:**

**long** time=System.*currentTimeMillis*();  
Date date=**new** Date(time);  
System.***out***.println(**"Normal date format : "**+ date);

**To get current value of the running JVM’’s high resolution time source, in nanoseconds:**

System.***out***.println(**"current time in nano seconds : "**+ System.*nanoTime*());

**To run the garbage collector:**

System.***gc();***

**To run finalization:**

System.***runFinalization();***

**To terminate the currently running JVM:** ( a non-zero status code indicates abnormal termination).

System.***exit(1);***

**To get all available currencies:**

Set<Currency> s=Currency.*getAvailableCurrencies*();  
System.***out***.println(s);

**To get the unique console object associated with current JVM:**

Console c = System.*console*();  
**if**(c != **null**)  
{  
 Currency currency = Currency.*getInstance*(Locale.***ITALY***);  
 c.printf(currency.getSymbol());  
 c.flush();  
}  
**else** System.***out***.println(**"No console attached"**);

**To read data from InputStream and to write data to OutputStream.**

FileInputStream IN = **new** FileInputStream(**new** File(**"C:\\Users\\sball\\Desktop\\customer.txt"**));  
FileOutputStream OUT = **new** FileOutputStream(**"system.txt"**);  
  
*// set input stream*System.*setIn*(IN);  
**char** c = (**char**) System.***in***.read();  
System.***out***.print(c);  
  
*// set output stream*System.*setOut*(**new** PrintStream(OUT));  
System.***out***.write(**"Hi Chakra\n"**.getBytes());  
  
*// set error stream*System.*setErr*(**new** PrintStream(OUT));  
System.***err***.write(**"Exception message\n"**.getBytes());