**Language Fundamentals**

1. Identifiers
2. Reserve words
3. Data types
4. Literals
5. Arrays
6. Type of Variables
7. Var-Arg methods
8. main()
9. Command line arguments
10. Java Coding standards

* **Identifiers:**

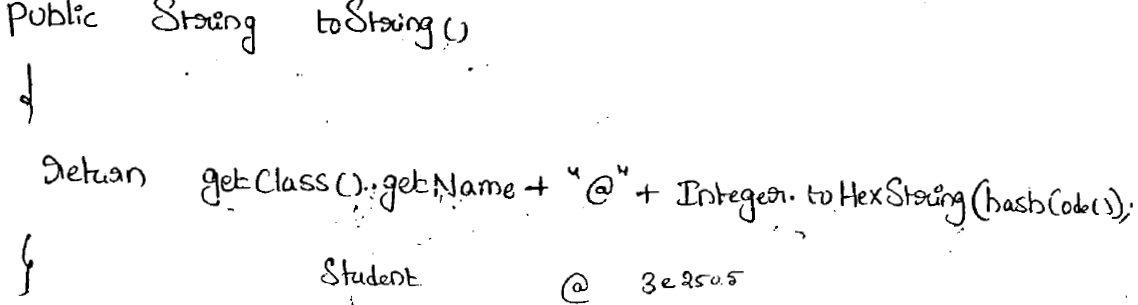
A name in Java program is called identifier, it can be class name or variable name or method name.

**Java.lang Package**

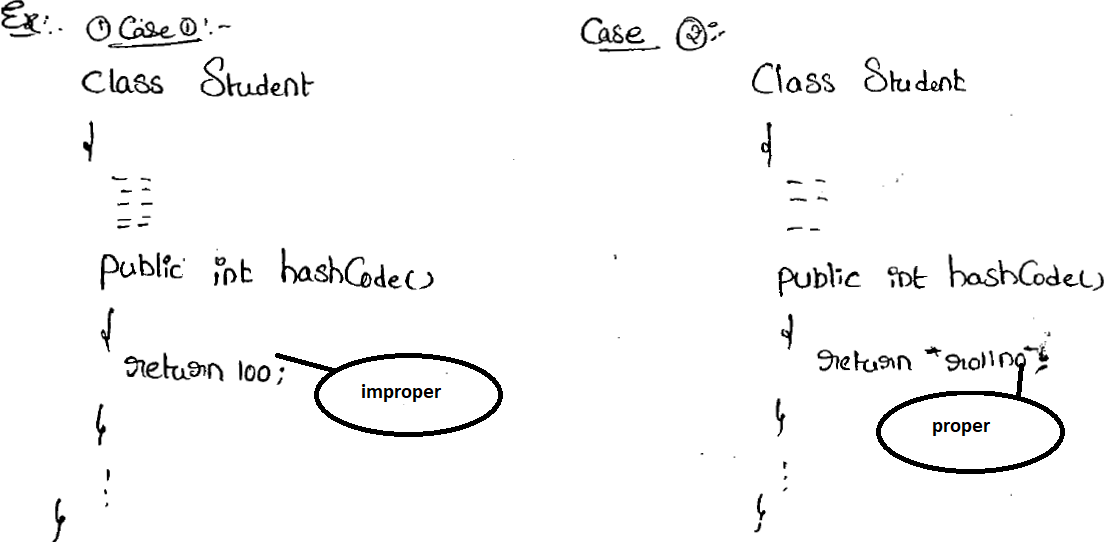
* **Object class defines the following 11 methods**

1. public string toString
2. public native int hascode()
3. public boolean equals(Object o)
4. protected native Object clone() throws ColneNotSupportedException
5. protected void finalize() throws Throwable
6. public final Class getClass()
7. public final void wait() throws InterruptedException
8. public final native void wait(long ms) throws InterruptedException
9. public final void wait(long ms, int ms) throws InterruptedException
10. public native final void notify()
11. public native final void notifyAll()

* **toString():**
* We can use toString() method to get String representation of an object.
* Whenever we are trying to print object reference internally toString() method will be called.
* If our class doesn’t contain toString() method then Object class toString() method will be called.
* Object class toString() method implies as follows:



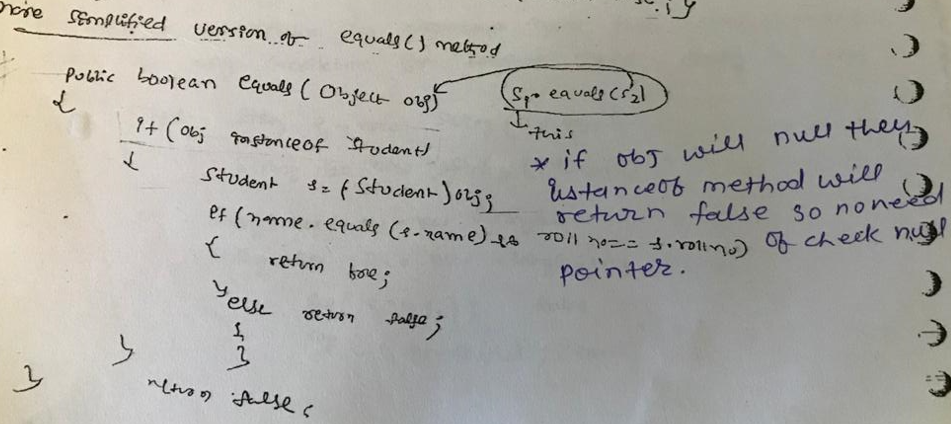
* In all wrapper class, in all Collection classes, String classes, StringBuffer, StringBuilder classes toString() method is overridden for meaningful String representation. Hence it is highly recommended to override toString method in our class also.
* **hascode():**
* For every object a unique number generated by JVM which is nothing but hascode().
* hascode() won’t represent address of Object
* JVM will use hascode while saving object into hashing related data structure like HashTable, HashMap, HasSet etc.
* The main advantage of saving object based on hascode is search operation will become easy (The most powerful search algorithm up to today is hashing)
* If we are giving chance object class hascode method it will generate hascode based on address of the object it doesn’t mean hascode represent address of the object.
* Based on our requirement we can override hascode method in our class to generate our own hascode.
* Override hasCode() method is said to be proper if for every object we have to generate a unique hascode.

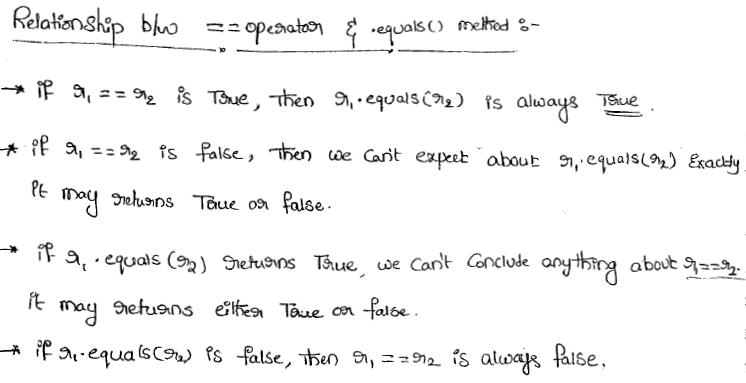


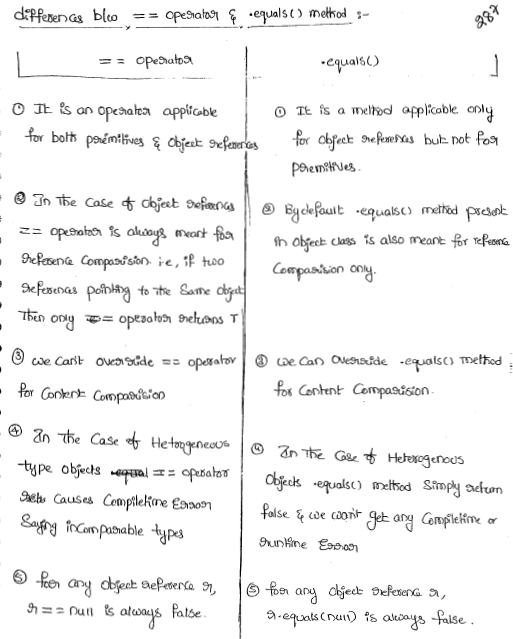
* **toString() vs hasCode()**

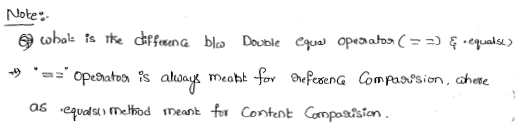
If we are giving chance to object class toString() method it will internally calls hasCode method. If we overriding toString() then our toString() method then our may not call hasCode().

* equals():
* We can use equals() method to check equality of two object. If our class does’t contain equals() method then object class equals() method will be exuted.









* Contract between .equals() method and hasCode() method:
* If two objects are equal by .equals() method then their hasCode() must be same be equal.
* If two object are not equals by .equals() method then there is no restriction on hasCode it may be return true or may be false.
* If hascode of two objects are equal then we can not conclude anything about .equals() method it may returns true or false.
* If hascode of two objects are not equal then these objects are always not equal by equals() method.

Note – To satisfied contract between equals and hasCode() method whenever we are overriding equals() method compulsory we have to override hascode() method otherwise we will not get any compile time error but is not a good programming proactive.

* clone()
* The process of creating exactly duplicate object is called cloning. Then main purpose of cloning is to maintain backup copy and to preserve state of an object.
* We can get cloned object by using clone() method of Object class.
* **Shallow cloning**
* The process of creating bitwise copy of an object is called shallow cloning.
* If the main object contains primitive variable, then exactly duplicate copies will be created in the cloned object.
* If the main object contains any reference variable then corresponding object will not be created just duplicate reference variable will be created pointing to old contained object.
* Object class clone() method meant for shallow cloning.
* For this cloning we will have to implements Cloneable interface and override clone() method in our cloneable class and it must return **super.clone() method.**
* **In this cloning reference will point to old object only and it will get change.**
* In shallowing cloning by using cloned object reference if we perform only change to the contained object then those changes will be reflected to the main object.
* Deep cloning:
* The process of creating exactly duplicate independent copy including contained object is called Deep cloning.