Naming Conventions

1. Variable names must be in Camel case starting with lower

   case. **Example:**

   audioSystem, myName,studentId

2. Names representing constants (final variables) must be all uppercase using underscore to separate words.

Example:

```
MAX_ITERATIONS, COLOR_RED
```

3. (i)Class based model name should normally use the Pascal

   Casing convention and end with “Model”. **Example:**

   class ProductModel

   (ii) Class based form name should be camel casing and end with “Form” like as signupForm, productForm.
   (iii) class based model attribute name should be start with “m_ “ like as m_productId, m_productName.
   (iv) Class based form attribute name should be start with “f_” like as f_productId,f_quantity.

4. Interface Name:Interface names should be use Camel Casing

   convention **Example:**

   interface <interface_name>{

   ```
   // declare constant fields
   ```

   }

5. Names representing methods must be verbs and written in Camel case starting with lower case.

Example:

```
getName(), computeTotalWidth()
```

6. Private class variables should have underscore (_) suffix.

Example:

```
class Person{

private String name_;
...
}
```

7. Arrays should be declared with their brackets next to the variable name.

Example:

```
double vertex[];
```

8.Function name should be like as login_page(), product_page()

9.App name should be like as app_login, app_product

Specific Naming Conventions

1. is prefix should be used for boolean variables and methods.

   Example:

   isSet, isVisible, isFinished, isFound, isOpen

2. Plural form should be used on names representing a collection of objects.

   Example:

   int values[];

3. n prefix should be used for variables representing a number of objects.

Example:

```
nPoints, nLines
```

4. No suffix should be used for variables representing an entity number.

   Example:

   tableNo, employeeNo

Exception Names:

Because exceptions should be classes, the class naming convention applies here. However, you should use the suffix “Error” on your exception names (if the exception actually is an error).

Types:

Type conversions must always be done explicitly. Never rely on implicit type conversion.

Example:

```
floatValue = (float) intValue;
```

Loops:

1. Loop control statements must be included in the for() or while() construction.

Example: for Loop:

```
sum = 0;
for (i = 0; i < 100; i++)

 sum += value[i];
```

```
while Loop:
boolean isDone = false;
while (!isDone) {

}
```

Layout:

1. Basic indentation should be 2.

Example:

```
for (i = 0; i < nElements; i++)
 a[i] = 0;
```

2. The if-else class of statements should have the following form:

Example:

```
if (condition)

 { statements; }

else
```

```
 { statements; }
```

3. A try-catch statement should have the following form:

Example:

```
try

 { statements; }

 catch (Exception exception)

 { statements; }

finally

{ statements; }
```

White Space

Operators should be surrounded by a space character.

- Reserved words should be followed by a white space.
- Commas should be followed by a white space.
- Colons should be surrounded by white space.
- Semicolons in for statements should be followed by a space character.

Example:

```
a = (b + c) * d;

while (true) {
doSomething (a, b, c, d);
case 100 :
for (i = 0; i < 10; i++) {
```

Variable Ordering :

Class variables order should be Public, Protected, Private.

Method Ordering:

Methods order should be Constructor , Public method , Protected method ,Private method.