**ASSIGNMENT 11**

**Q1.What is Clean Code**

So, before we jump into the details of clean code. lets understand what do we mean by clean code. Honestly, there can not be one good answer to this. In programming, some concerns reach across and hence result in general principle.But then every programming langauge and paradigm present their own set of nuances, which mandates us to adopt befitting pratices. Broadly,Clean code can be summarized as a code that any developers can read and change easily.

"Any Fool can write code that a computer can understand.Good programmers write code that humans can understand"

**Q2.Why Should we care about clean code?**

Writing Clean Code is a matter of personal habit as much as it's a matter of skill. As a developer, we grow through experience and knowledge over time. But, we must ask why should invest in developing clean code after all?we get that others will probably find it easier to read our code.

**1)Magic Number**

A magic number means we are assigning a number with no clear meaning. Sometimes we use a value for a specific purpose, and we don't assign the value in a meaningful variable. The problem is that when someone works with your code, then the person doesn't know the meaning of that direct value.

Example

//Bad Practice

For(int i=0;i<50;i++)

{

//do something

}

//Good Practice  
int Number\_Of\_Iteration=50;

For(int i=0;i<Number\_Of\_Iteration;i++)  
{

//Do something

}

**2)Comments**

This one is a kind of personal attack on someone. Comments help people to understand later, and they help other programmers to work on the same project. Comments in code mean maybe your code is not self-explanatory.Comments should be good, but your code needs to be self-explanatory

//this loop print 0 to 40 value

for(int i=0;i<40;i++)  
{

System.out.println(i);

}

**3) Avoid Large Functions**

When a function or a class is much larger, then it is suggested to separate it into multiples. This will make our code easier, clean, easy to understand, and also reusable.

Suppose we need to add and subtract two numbers. We can do it with a single function. But the good practice is to divide them into two. When there are individual functions, then this will be reusable in the whole application.

//bad code

class Operation{

public String AddSub(int a,int b){

final int add=a+b;

final int sum=a-b;

return a+" "+b;

}  
}

//good code

class Operation

{

public String Add(int a,int b)  
 {

return a+b;  
 }

public String Sub(int a,int b){

return a-b;

}  
}

**4 Code Repetation**

Repeated code means a code block that is repeated in your code more than once. This means your code portion needs to be extracted into a function.

//bad code

for(int i=0;i<10;i++){

int a=i\*i\*i;

System.out.println(a);

}

//good code

int a=0;

for(int i=0;i<10;i++)

{

a=i\*i\*i;

System.out.println(a);

}

**5 Variable Name**

Camel case is the naming standard for both variables and functions, and also other identifiers. This means a name is supposed to begin with a small letter, and every first letter of the next word will be uppercase.

int SetAddResult=0;

public int AddTwoValue(int a,int b)

{

SetAddResult=a+b;

}

**6 Meaning Full Name**

A meaningful name is one of the most important conventions. Always use a meaningful name for variables, functions, and others. Choose a name that expresses the meaning of your purpose.

If we need a function that will get the user's bank information, then we must not use a name like getUserInfo or something like that. We should use getUserBankInfo to be more specific

**7 Favor Descriptive Over Concise**

Try to use detail for any naming. Suppose we need a function that will find a user with their phone. Here we can use meaningful names, but there is a huge possibility of mistakes if there are other, similar functions.

We must use a detailed, meaningful name that expresses the meaning in a nutshell.

//We want a function for search user against phone no

//Bad Code

public void SearchUser(int PhoneNumber)  
 {

//do something  
 }

//Good Code

public void SearchUserByPhoneNumber(int PhoneNumber)

{

//Do something

}

**8. Use Nouns for Class Name & Use Pascal Case**

Classes don't take things; they are the things. Class is mainly a blueprint for something. Don't use the verb in the class name.Also, a class should contain Pascal case. Camel case is used for objects, so this won't be very clear if you use camel case for class.

//bad practice

class MakeCar = {

//...

}

//Good practice

class Car = {

//...

}

**9. Capitalize Constant Values (SNAKE UPPER CASE)**

This is another convention that we need to follow. Always use fully capitalized names for constants.Snake uppercase means all the letters will be uppercase, and an underscore will separate all the words.

final int DAYS\_IN\_A\_YEAR=365;

**10)Avoid One-Letter Variable Names**

A one-letter variable is a very, very bad thing to use. Don't use this for a variable name.

But in a loop, we use some variables with a letter, which is OK to use.

//bad practice

const q = () => {

//....

}

//good practice

const query = () => {

//....

}