25/06/2020

AKSHATHA.Y.
4AL18EC005

• Report :
 • Inheritance * polymorphism.
 * Inheritance :
  Inheritance is one of the most importan concepts oriented programming.

```
class mother
{
 public
 mother() {};
 void say Hi(){
  cout<<"Hi";
  }
 }
class Daughter
{
 public;
 Daughter(){};
 }
```

• Protected members:

```
class mother {
 public;
 void say Hi (){
 cout<<var;
  }
 private ;
 int var=0;
 protected;
 int some var;
 };
```

- Class Constructor and Destructor

```
class Mother {
public:
Mother ()
{
cout<<"Mother ctor" << endl;
}
~Mother ()
{cout <<"Mother dtor"<< endl;
}
};
```

- Polymorphism:

```
class Enemy {
protected
int attack power;
public:
void set Attack power (int a) {
 attack power = a;
 }
 };
```

- Templates, exception and Files:

Function Templates.

```
int sum (int a, int b){
return a+b;
}
 int main () {
 int x=4 , y= 15;
 cout << sum (x ,y) << endl;
 }
```

- Abstract classes:

```
class enemy{
public;
virtual void attack(){
cout<< "Enemy"<<endl;
}
};
class Ninja; public enemy {
public;
void attack(){
cout<< "Ninja!"<<endl;
}
};
class moster: public enemy{
public:
void attack(){
cout<< "Monster!"<<endl;
}
};
```

- Function templates with multiple parameters

```
template <class T. class U>
T smaller (T a, U b){
  return (a<b ? a:b);
  }
```

$\boxed{\text{template <class T. class U>}}$

Notes

Work to do

• Class Templates

```
template <class T >
class pair {
private.
 T first , second;
 public;
 pair (Ta, Tn);
 first (a) , second (b){
}
3;
```

• Template specialization.

```
template <classT>
class Myclass {
public
Myclass (T x ) {
cout << x<< " - not a char" << endl;
}
}
template< >
class myclass < char>{
public;
my class (char x ) {
cout<<x<< " Is a char! " <<endl;
}
};
```

- Exceptions:

T W T F S S M T W T F S S M