```cpp
//
// Created by 15165 on 2026/1/14.
//

#ifndef WORKSHOP_2_GPUNODE_H
#define WORKSHOP_2_GPUNODE_H
#include <iostream>
#include <string>

class GPUNODE
{
public://accessable outside the function
    //constructors
    // 3types of constructor
    GPUNODE();//defult
    GPUNODE(double, long, short, long, std::string); //parameter based constructor
    GPUNODE(const GPUNODE&); //copy constructor, create object by existing object, we don't want to make a copy, we want to save memory space, so we want to make it a constant
    void setModel(double val);
    double getModel() const;

private://not accessable outside the function
    double model;
    long speed;
    short node;
    long throughput;
    std::string color;
    //position pending
};

///Definition
inline GPUNODE::GPUNODE()
    :model(0),speed(0),node(-1),throughput(-1),color("cyan") //member initialization list
{

}

```

```cpp
inline GPUNODE::GPUNODE(double m, long s, short n, long t, std::string col)
    :model(m), speed(s), node(n), throughput(t), color(col)
{

}

inline  GPUNODE::GPUNODE(const GPUNODE& other)
    :model(other.model),speed(other.speed),node(other.node),throughput(other.throughput),color(other.color)
{

}

inline void GPUNODE::setModel(double val)
{
    model = val;
}

inline double GPUNODE::getModel() const
{
    return model;
}


inline void testGPUNode() //outside the scope of class definition, can only access to public segment
{
    GPUNODE item; //item is an object instance; //calling construction by the compiler
    item.setModel(37.5);
    //item.model; not accessable
    std::cout << "item 1: " << item.getModel() << std::endl;

    GPUNODE item2;
    item2.setModel(40.5);
    std::cout << "item 2: " << item2.getModel() << std::endl;

    GPUNODE cloneGPU(item);
    item.setModel(36);

    std::cout << "item 3: " << item.getModel() << std::endl;
    std::cout << "clone: " << cloneGPU.getModel() << std::endl;

}
#endif //WORKSHOP_2_GPUNODE_H
```