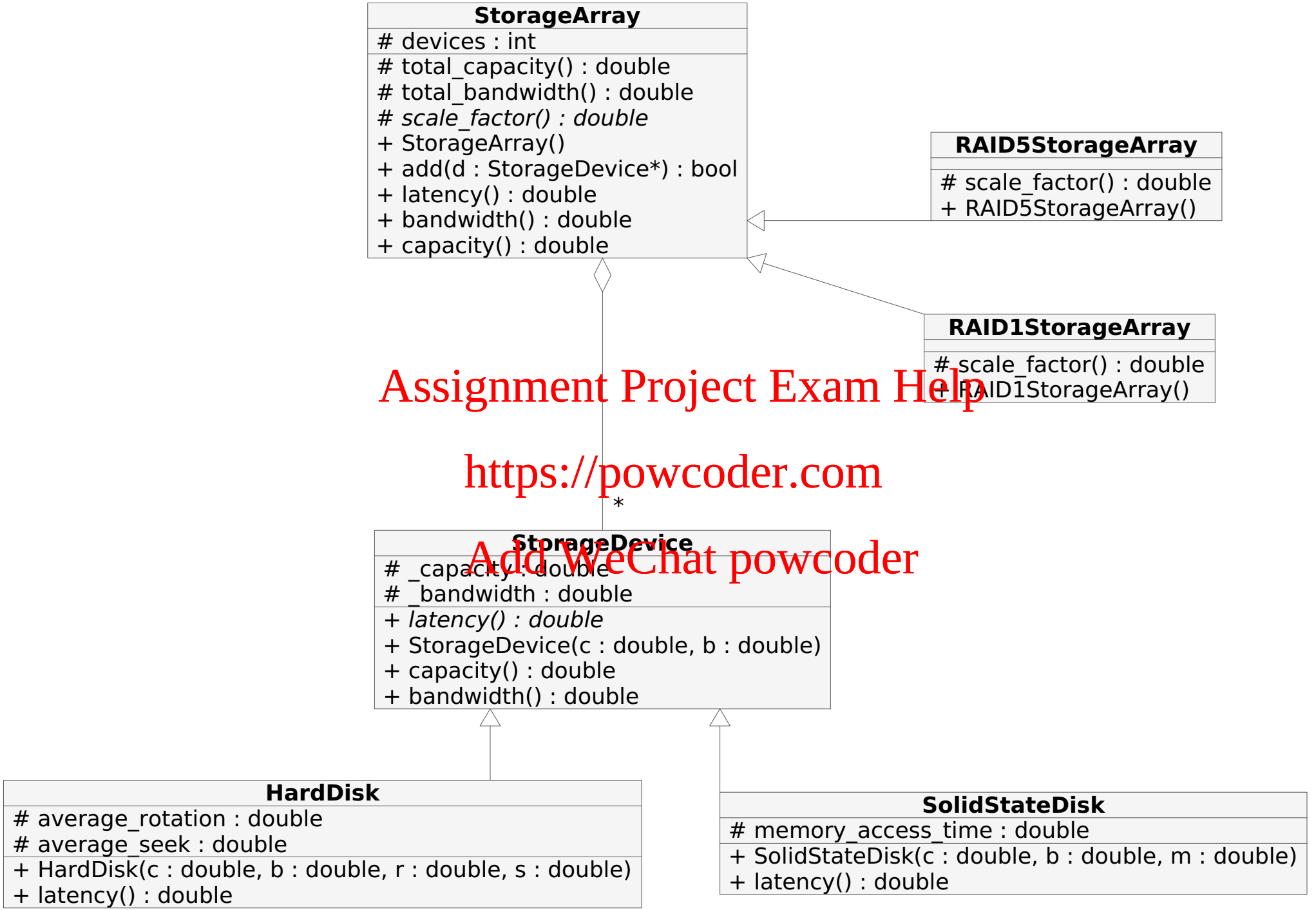
StorageArray
devices : int
total_capacity() : double
total_bandwidth() : double
scale_factor() : double
+ StorageArray()
+ add(d : StorageDevice*) : bool
+ latency() : double
+ bandwidth() : double
+ capacity() : double
RAID5StorageArray
scale_factor() : double
+ RAID5StorageArray()
RAID1StorageArray
scale_factor() : double
+ RAID1StorageArray()
Assignment Project Exam Help
https://powcoder.com
*
StorageDevice
_capacity : double
_bandwidth : double
+ latency() : double
+ StorageDevice(c : double, b : double)
+ capacity() : double
+ bandwidth() : double
Add WeChat powcoder
HardDisk
average_rotation : double
average_seek : double
+ HardDisk(c : double, b : double, r : double, s : double)
+ latency() : double
SolidStateDisk
memory_access_time : double
+ SolidStateDisk(c : double, b : double, m : double)
+ latency() : double

```
#include <iostream>

using namespace std;

class StorageDevice {
protected:
  double _capacity, _bandwidth;
public:
  virtual double latency() = 0;
  StorageDevice(double c, double b) {
    _bandwidth = b;
    _capacity = c;
  }
  double capacity() const { return _capacity; }
  double bandwidth() const { return _bandwidth; }
};

class HardDisk : public StorageDevice {
protected:
  double average_rotation, average_seek;
public:
  HardDisk(double c, double b, double r, double s) : StorageDevice (c,b) {
    average_rotation = r;
    average_seek = s;
  }
  double latency() {
    return average_rotation + average_seek;
  }
};

class SolidStateDisk : public StorageDevice {
protected:
  double memory_access_time;
public:
  SolidStateDisk(double c, double b, double m) : StorageDevice (c,b) {
    memory_access_time = m;
  }
  double latency() {
    return memory_access_time;
  }
};

class StorageArray {
protected:
  int devices;
  StorageDevice *device[12];

  double total_capacity() const {
    double sum = 0;
    for (int n=0; n<devices; n++)
      sum += device[n]->capacity();
    return sum;
  }

  double total_bandwidth() const {
    double sum = 0;
    for (int n=0; n<devices; n++)
      sum += device[n]->bandwidth();
    return sum;
  }

  virtual double scale_factor() const = 0;
```



```
public:
  StorageArray() {
    devices = 0;
  }
  bool add(StorageDevice *d) {
    if (devices >= 12)
      return false;
    device[devices++] = d;
    return true;
  }

  double latency() {
    if (devices < 1)
      return 0;
    double max = device[0]->latency();
    for (int n=1; n<devices; n++)
      if (device[n]->latency() > max)
        max = device[n]->latency();
    return max;
  }

  double bandwidth() const {
    return total_bandwidth()*scale_factor();
  }

  double capacity() const {
    return total_capacity()*scale_factor();
  }
};

class RAID1StorageArray : public StorageArray {
protected:
  double scale_factor() const { return 0.5; }
public:
  RAID1StorageArray() : StorageArray() { }
};

class RAID5StorageArray : public StorageArray {
protected:
  double scale_factor() const { return ((double) devices)/(devices - 1); }
public:
  RAID5StorageArray() : StorageArray() { }
};

int main() {

  RAID1StorageArray raid1;
  RAID5StorageArray raid5;

  for (int n=0; n<4; n++) {
    raid1.add(new HardDisk(500,100,4,8));
    raid5.add(new SolidStateDisk(30,300,0.01));
  }

  cout << "Bandwidth of RAID1 system is " << raid1.bandwidth() << endl;
  cout << "Latency of RAID1 array is " << raid1.latency() << endl;
  cout << "Effective capacity of RAID5 system is " << raid5.capacity() << endl;

  return 0;
}
```