|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Line** | **Code** | **Wc** | **Wn** | **Wi** | **Wio** | **Wdi** | **Wf** | **We** | **Wp** | **Wa** |
|  | **public** **interface** Shape { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** draw(); | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
|  | **public** **double** calculateArea(); | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 |
|  | **public** **double** calculatePerimeter(); | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **abstract** **class** TwoDShape **implements** Shape{ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **private** String name; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **private** **int** length; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **private** **double** radius; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **private** **int** width; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **public** TwoDShape(String name, **int** length, **int** width) { | 0 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 1 |
|  | **this**.name = name; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **this**.length = length; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **this**.width = width; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Encapsulation due to getters and setters | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** String getName() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** name; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setName(String name) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.name = name; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **int** getLength() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** length; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setLength(**int** length) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.length = length; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **double** getRadius() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** radius; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setRadius(**double** radius) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.radius = radius; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **int** getWidth() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** width; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setWidth(**int** width) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.width = width; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **abstract** **class** ThreeDShape **implements** Shape{ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **private** String name; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **private** **int** length; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **private** **int** breadth; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **private** **int** width; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | //Encapsulation due to getters and setters | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** ThreeDShape(String name, **int** length, **int** breadth, **int** width) { | 0 | 0 | 2 | 0 | 0 | 5 | 0 | 0 | 1 |
|  | **this**.length = length; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **this**.breadth = breadth; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **this**.width = width; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | **this**.name = name; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** String getName() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** name; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setName(String name) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.name = name; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **int** getLength() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** length; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setLength(**int** length) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.length = length; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **int** getBreadth() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** breadth; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setBreadth(**int** breadth) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.breadth = breadth; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **int** getWidth() { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **return** width; | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** setWidth(**int** width) { | 0 | 0 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
|  | **this**.width = width; | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **abstract** **double** calculateVolume(); | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 1 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **class** Square **extends** TwoDShape **implements** Shape{ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** Square(String name, **int** length) { | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 |
|  | **super**(name, length, length); | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** draw() { | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 1 | 0 |
|  | System.***out***.println("Drawing Square"); | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **double** calculateArea() { | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 1 | 0 |
|  | **double** area = **super**.getLength() \* **super**.getWidth(); | 0 | 0 | 3 | 0 | 2 | 4 | 1 | 1 | 0 |
|  | **return** area; | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **double** calculatePerimeter() { | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 1 | 0 |
|  | **double** perimeter = 4 \* **super**.getLength() ; | 0 | 0 | 3 | 0 | 2 | 2 | 1 | 1 | 0 |
|  | **return** perimeter; | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **class** Cube **extends** ThreeDShape **implements** Shape{ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** Cube(String name, **int** length) { | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 |
|  | **super**(name, length, length, length); | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **double** calculateArea() { | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 1 | 0 |
|  | **double** area = 6 \* (**super**.getLength() \* **super**.getLength()); | 0 | 0 | 3 | 0 | 2 | 4 | 1 | 1 | 0 |
|  | **return** area; | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **double** calculatePerimeter() { | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 1 | 0 |
|  | **double** perimeter = 12 \* **super**.getLength(); | 0 | 0 | 3 | 0 | 2 | 2 | 1 | 1 | 0 |
|  | **return** perimeter; | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** draw() { | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 1 | 0 |
|  | System.***out***.println("Drawing Cube"); | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Polymorphism | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | @Override | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **double** calculateVolume() { | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 1 | 0 |
|  | **double** volume = **super**.getLength() \* **super**.getBreadth() \* **super**.getWidth(); | 0 | 0 | 3 | 0 | 2 | 6 | 1 | 1 | 0 |
|  | **return** volume; | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **class** PlusThread **implements** Runnable { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | Object lock; | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 |
|  | **int** start; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
|  | **int** range; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
|  | **public** PlusThread(Object lock, **int** start, **int** range) { | 0 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 0 |
|  | **this**.lock = lock; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
|  | **this**.start = start; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
|  | **this**.range = range; | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** addNumbers(Object lock, **int** start, **int** range) **throws** InterruptedException { | 1 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 0 |
|  | Thread.*currentThread*().setName("Thread-0"); | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **for** (**int** i = start; i <= range; i++) { | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | lock.notify(); | 0 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | System.***out***.println(Thread.*currentThread*().getName() + "=> " + i + "+" + i + "=" + (i + i)); | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | Thread.*sleep*(1000); | 0 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | lock.wait(); | 0 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **void** run() { | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **synchronized**(lock){ | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **try** { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **this**.addNumbers(lock, start, range); | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | } **catch** (InterruptedException e) { | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | // **TODO** Auto-generated catch block | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | e.printStackTrace(); | 0 | 0 | 2 | 1 | 0 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **import** java.util.\*; | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **class** Main { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **static** **void** main(String[] args) { | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
|  | Scanner sc = **new** Scanner(System.***in***); | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
|  | //Variable declaration and instantiation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **int** length1 = 0; | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
|  | **int** length2 = 0; | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
|  | ArrayList<Cube> al = **new** ArrayList<>(); | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
|  | **try** { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.print("Enter Square details "); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.print("Enter length of a side of a square : "); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | length1 = sc.nextInt(); | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|  | System.***out***.print("Enter Cube details "); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("Enter length of a side of a cube : "); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | length2 = sc.nextInt(); | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|  | }**catch**(InputMismatchException e) { | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | e.printStackTrace(); | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Creating a Square object | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | Square sq = **new** Square("Square1",length1); | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
|  | //Creating 10 cube objects | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **for**(**int** i = 0; i<10;i++) { | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | al.add(**new** Cube("Cube"+i,length2)); | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Calculate area of square | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **double** sArea = sq.calculateArea(); | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | System.***out***.println("The area of the square = " + sArea); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | //Check square size based on the area | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **if**(sArea>1000.0) { | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("This is a very large square"); | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | }**else** **if**(sArea > 500.0 && sArea < 900.00) { | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("This is a medium square."); | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | }**else** **if**((sArea > 100.0 && sArea < 300.00) || (sArea > 300.0 && sArea < 500.00) ) { | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("This is a small square."); | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | }**else** { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("This is a very small square."); | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Calculate volume of cube | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **double** cVolume = al.get(0).calculateVolume(); | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 |
|  | System.***out***.println("The volume of the cube = " + cVolume); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | //Guess a number | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("Enter a number between 1 to 5, to check whether you are lucky"); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **int** num = sc.nextInt(); | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
|  | //While loop | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **while**(num < 6) { | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **switch**(num) { | 6 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **case** 1: System.***out***.println("Congratulations, You are lucky"); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **case** 2: System.***out***.println("You are unlucky"); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **case** 3: System.***out***.println("You are unlucky"); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **case** 4: System.***out***.println("You are unlucky"); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **case** 5: System.***out***.println("You are unlucky"); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **default**:System.***out***.println("Invalid input"); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("Enter a number to get the factorial : "); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **int** fac = sc.nextInt(); | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
|  | //Get factorial of a number using a recursive method | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("Factorial = " + *factorial*(fac)); | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 0 |
|  | //Printing a star pattern using a nested for loop | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("Printing a Pattern"); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | **for**(**int** i=0; i<10; i++){ | 2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|  | **for**(**int** j=0; j<=i; j++){ | 2 | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|  | System.***out***.print("\* "); | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println(); | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | System.***out***.println("Adding numbers using Threads"); | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  | Object lock = **new** Object(); | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
|  | //Invocation of a Thread | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | PlusThread plusThread = **new** PlusThread(lock,2,10); | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 |
|  | Thread thread = **new** Thread(plusThread); | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
|  | thread.start(); | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | //Recursive method | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **public** **static** **double** factorial(**int** num) { | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 |
|  | **if**(num == 0) { | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **return** 1; | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | }**else** { | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **return** (num \* *factorial*(num-1)); | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **Total** | **28** | **38** | **207** | **32** | **76** | **130** | **37** | **19** | **53** |
|  |  | **Wc** | **Wn** | **Wi** | **Wio** | **Wdi** | **Wf** | **We** | **Wp** | **Wa** |

|  |
| --- |
| **public** **interface** Shape { |
|  |
| **public** **void** draw(); |
| **public** **double** calculateArea(); |
| **public** **double** calculatePerimeter(); |
|  |
| } |
| **public** **abstract** **class** TwoDShape **implements** Shape{ |
|  |
| **private** String name; |
| **private** **int** length; |
| **private** **double** radius; |
| **private** **int** width; |
|  |
| **public** TwoDShape(String name, **int** length, **int** width) { |
| **this**.name = name; |
| **this**.length = length; |
| **this**.width = width; |
| } |
| //Encapsulation due to getters and setters |
| **public** String getName() { |
| **return** name; |
| } |
| **public** **void** setName(String name) { |
| **this**.name = name; |
| } |
| **public** **int** getLength() { |
| **return** length; |
| } |
| **public** **void** setLength(**int** length) { |
| **this**.length = length; |
| } |
| **public** **double** getRadius() { |
| **return** radius; |
| } |
| **public** **void** setRadius(**double** radius) { |
| **this**.radius = radius; |
| } |
| **public** **int** getWidth() { |
| **return** width; |
| } |
| **public** **void** setWidth(**int** width) { |
| **this**.width = width; |
| } |
| } |
| **public** **abstract** **class** ThreeDShape **implements** Shape{ |
| **private** String name; |
| **private** **int** length; |
| **private** **int** breadth; |
| **private** **int** width; |
| //Encapsulation due to getters and setters |
| **public** ThreeDShape(String name, **int** length, **int** breadth, **int** width) { |
| **super**(); |
| **this**.length = length; |
| **this**.breadth = breadth; |
| **this**.width = width; |
| **this**.name = name; |
| } |
| **public** String getName() { |
| **return** name; |
| } |
| **public** **void** setName(String name) { |
| **this**.name = name; |
| } |
| **public** **int** getLength() { |
| **return** length; |
| } |
| **public** **void** setLength(**int** length) { |
| **this**.length = length; |
| } |
| **public** **int** getBreadth() { |
| **return** breadth; |
| } |
| **public** **void** setBreadth(**int** breadth) { |
| **this**.breadth = breadth; |
| } |
| **public** **int** getWidth() { |
| **return** width; |
| } |
| **public** **void** setWidth(**int** width) { |
| **this**.width = width; |
| } |
| **public** **abstract** **double** calculateVolume(); |
| } |
| **public** **class** Square **extends** TwoDShape **implements** Shape{ |
|  |
| **public** Square(String name, **int** length) { |
| **super**(name, length, length); |
| } |
| //Polymorphism |
| @Override |
| **public** **void** draw() { |
| System.***out***.println("Drawing Square"); |
| } |
| //Polymorphism |
| @Override |
| **public** **double** calculateArea() { |
| **double** area = **super**.getLength() \* **super**.getWidth(); |
| **return** area; |
| } |
| //Polymorphism |
| @Override |
| **public** **double** calculatePerimeter() { |
| **double** perimeter = 4 \* **super**.getLength() ; |
| **return** perimeter; |
| } |
| } |
| **public** **class** Cube **extends** ThreeDShape **implements** Shape{ |
|  |
| **public** Cube(String name, **int** length) { |
| **super**(name, length, length, length); |
| } |
| //Polymorphism |
| @Override |
| **public** **double** calculateArea() { |
| **double** area = 6 \* (**super**.getLength() \* **super**.getLength()); |
| **return** area; |
| } |
| //Polymorphism |
| @Override |
| **public** **double** calculatePerimeter() { |
| **double** perimeter = 12 \* **super**.getLength(); |
| **return** perimeter; |
| } |
| //Polymorphism |
| @Override |
| **public** **void** draw() { |
| System.***out***.println("Drawing Cube"); |
| } |
| //Polymorphism |
| @Override |
| **public** **double** calculateVolume() { |
| **double** volume = **super**.getLength() \* **super**.getBreadth() \* **super**.getWidth(); |
| **return** volume; |
| } |
| } |
| **public** **class** PlusThread **implements** Runnable { |
|  |
| Object lock; |
| **int** start; |
| **int** range; |
| **public** PlusThread(Object lock, **int** start, **int** range) { |
| **this**.lock = lock; |
| **this**.start = start; |
| **this**.range = range; |
| } |
| **public** **void** addNumbers(Object lock, **int** start, **int** range) **throws** InterruptedException { |
| Thread.*currentThread*().setName("Thread-0"); |
| **for** (**int** i = start; i <= range; i++) { |
| lock.notify(); |
| System.***out***.println(Thread.*currentThread*().getName() + "=> " + i + "+" + i + "=" + (i + i)); |
| Thread.*sleep*(1000); |
| lock.wait(); |
| } |
| } |
| **public** **void** run() { |
| **synchronized**(lock){ |
| **try** { |
| **this**.addNumbers(lock, start, range); |
| } **catch** (InterruptedException e) { |
| // **TODO** Auto-generated catch block |
| e.printStackTrace(); |
| } |
| } |
| } |
| } |
| **import** java.util.\*; |
| **public** **class** Main { |
| **public** **static** **void** main(String[] args) { |
| Scanner sc = **new** Scanner(System.***in***); |
| //Variable declaration and instantiation |
| **int** length1 = 0; |
| **int** length2 = 0; |
| ArrayList<Cube> al = **new** ArrayList<>(); |
| **try** { |
| System.***out***.print("Enter Square details "); |
| System.***out***.print("Enter length of a side of a square : "); |
| length1 = sc.nextInt(); |
|  |
| System.***out***.print("Enter Cube details "); |
| System.***out***.println("Enter length of a side of a cube : "); |
| length2 = sc.nextInt(); |
| }**catch**(InputMismatchException e) { |
| e.printStackTrace(); |
| } |
| //Creating a Square object |
| Square sq = **new** Square("Square1",length1); |
| //Creating 10 cube objects |
| **for**(**int** i = 0; i<10;i++) { |
| al.add(**new** Cube("Cube"+i,length2)); |
| } |
| //Calculate area of square |
| **double** sArea = sq.calculateArea(); |
| System.***out***.println("The area of the square = " + sArea); |
| //Check square size based on the area |
| **if**(sArea>1000.0) { |
| System.***out***.println("This is a very large square"); |
| }**else** **if**(sArea > 500.0 && sArea < 900.00) { |
| System.***out***.println("This is a medium square."); |
| }**else** **if**((sArea > 100.0 && sArea < 300.00) || (sArea > 300.0 && sArea < 500.00) ) { |
| System.***out***.println("This is a small square."); |
| }**else** { |
| System.***out***.println("This is a very small square."); |
| } |
| //Calculate volume of cube |
| **double** cVolume = al.get(0).calculateVolume(); |
| System.***out***.println("The volume of the cube = " + cVolume); |
| //Guess a number |
| System.***out***.println("Enter a number between 1 to 5, to check whether you are lucky"); |
| **int** num = sc.nextInt(); |
| //While loop |
| **while**(num < 6) { |
| **switch**(num) { |
| **case** 1: System.***out***.println("Congratulations, You are lucky"); |
| **case** 2: System.***out***.println("You are unlucky"); |
| **case** 3: System.***out***.println("You are unlucky"); |
| **case** 4: System.***out***.println("You are unlucky"); |
| **case** 5: System.***out***.println("You are unlucky"); |
| **default**:System.***out***.println("Invalid input"); |
| } |
| } |
| System.***out***.println("Enter a number to get the factorial : "); |
| **int** fac = sc.nextInt(); |
| //Get factorial of a number using a recursive method |
| System.***out***.println("Factorial = " + *factorial*(fac)); |
| //Printing a star pattern using a nested for loop |
| System.***out***.println("Printing a Pattern"); |
| **for**(**int** i=0; i<10; i++){ |
| **for**(**int** j=0; j<=i; j++){ |
| System.***out***.print("\* "); |
| } |
| System.***out***.println(); |
| } |
| System.***out***.println("Adding numbers using Threads"); |
| Object lock = **new** Object(); |
| //Invocation of a Thread |
| Thread plusThread = **new** Thread(**new** PlusThread(lock,2,10)); |
| plusThread.start(); |
| } |
| //Recursive method |
| **public** **static** **double** factorial(**int** num) { |
| **if**(num == 0) { |
| **return** 1; |
| }**else** { |
| **return** (num \* *factorial*(num-1)); |
| } |
| } |
| } |

System.***out***.println("Adding numbers using Threads");

Object lock = **new** Object();

//Invocation of a Thread

PlusThread plusThread = **new** PlusThread(lock,2,10);

Thread thread = **new** Thread(plusThread);

thread.start();