

```

graph LR
    ClientC[ClientC] --> Broker[Broker]
    Broker --> ServerA[ServerA: DateRetriever]
    Broker --> ServerB[ServerB: TimeRetriever]
  
```

```

classDiagram
    class Client {
        +main(args: String[])
    }
    class Broker {
        <<interface>>
        +registerServer(serverName: String, remoteHost: String)
        +executeService(serviceName: String, serviceParameters: List<Object>): Object
        +registerService(serverName: String, serviceName: String, serviceParameters: Map<String, String>, returnType: String)
        +terminateService(serverName: String, serviceName: String)
        +listServices(): List<Service>
    }
    class BrokerImpl {
        -serversMap: Map<String, String>
        -servicesList: List<Service>
        +registerServer(serverName: String, remoteHost: String)
        +executeService(serviceName: String, serviceParameters: List<Object>): Object
        +registerService(serverName: String, serviceName: String, serviceParameters: Map<String, String>, returnType: String)
        +terminateService(serverName: String, serviceName: String)
        +listServices(): List<Service>
    }
    class Service {
        -serviceName: String
        -serviceParameters: Map<String, String>
        -returnType: String
        -serverName: String
        +getServiceName(): String
        +getServiceParameters(): Map<String, String>
        +getReturnType(): String
        +getServerName(): String
    }
    class ServiceProvider {
        <<interface>>
        +getDate(): String
        +getCompleteDate(): String
        +getDateWithArgs(name: String, age: int): String
    }
    class DateRetriever {
        <<interface>>
    }
    class DateRetrieverImpl {
        +getDate(): String
        +getCompleteDate(): String
        +getDateWithArgs(name: String, age: int): String
    }
    class TimeRetriever {
        <<interface>>
        +getTime(): String
        +getTimeWithArgs(name: String, age: int, weight: int): String
    }
    class TimeRetrieverImpl {
        +getTime(): String
        +getTimeWithArgs(name: String, age: int, weight: int): String
    }

    Client "0..*" -- "0..*" Broker
    Broker <|-- BrokerImpl
    Broker "0..*" -- "0..*" ServiceProvider
    Broker "0..*" -- "0..*" Service
    BrokerImpl "1" -- "0..*" Service
    ServiceProvider "1" -- "1" DateRetriever
    ServiceProvider "1" -- "1" TimeRetriever
    DateRetriever <|-- DateRetrieverImpl
    TimeRetriever <|-- TimeRetrieverImpl
  
```

The diagram illustrates a Broker-based architecture. At the top, a 'Client' is connected to a 'Broker' via a single line. The 'Broker' is represented as a horizontal bar containing four square nodes. Below the 'Broker', there are two groups of 'DataRetriever' and 'TimerRetriever' components. Each group consists of a square node connected to a stack of three rectangles. The 'DataRetriever' group has two such nodes, and the 'TimerRetriever' group has two. Lines connect the 'Broker' nodes to the retriever nodes: the first two 'Broker' nodes connect to the first 'DataRetriever' node, the third 'Broker' node connects to the second 'DataRetriever' node, and the fourth 'Broker' node connects to the first 'TimerRetriever' node. The 'Client' is also connected to the 'Broker' via a line that passes through a small circle.