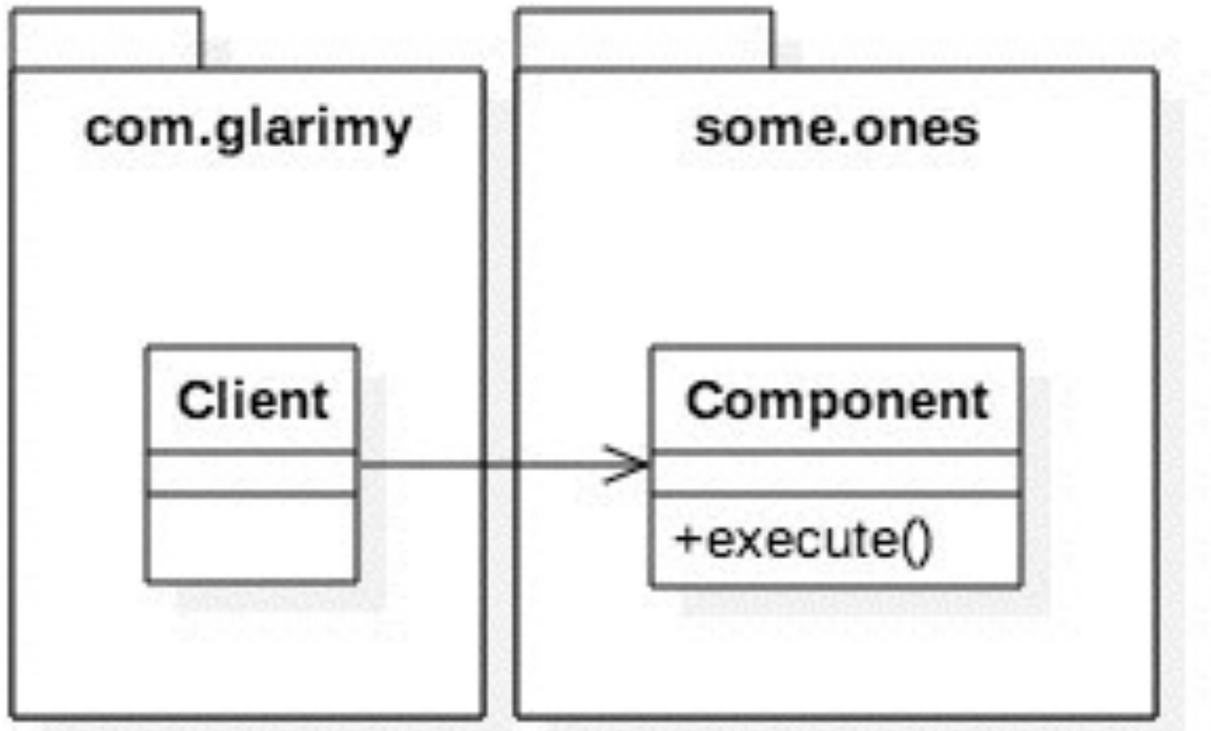
ADAPTER

Krishna Mohan Koyya
krishna@glarimy.com | www.glarimy.com

ADAPTER – PROBLEM ILLUSTRATION

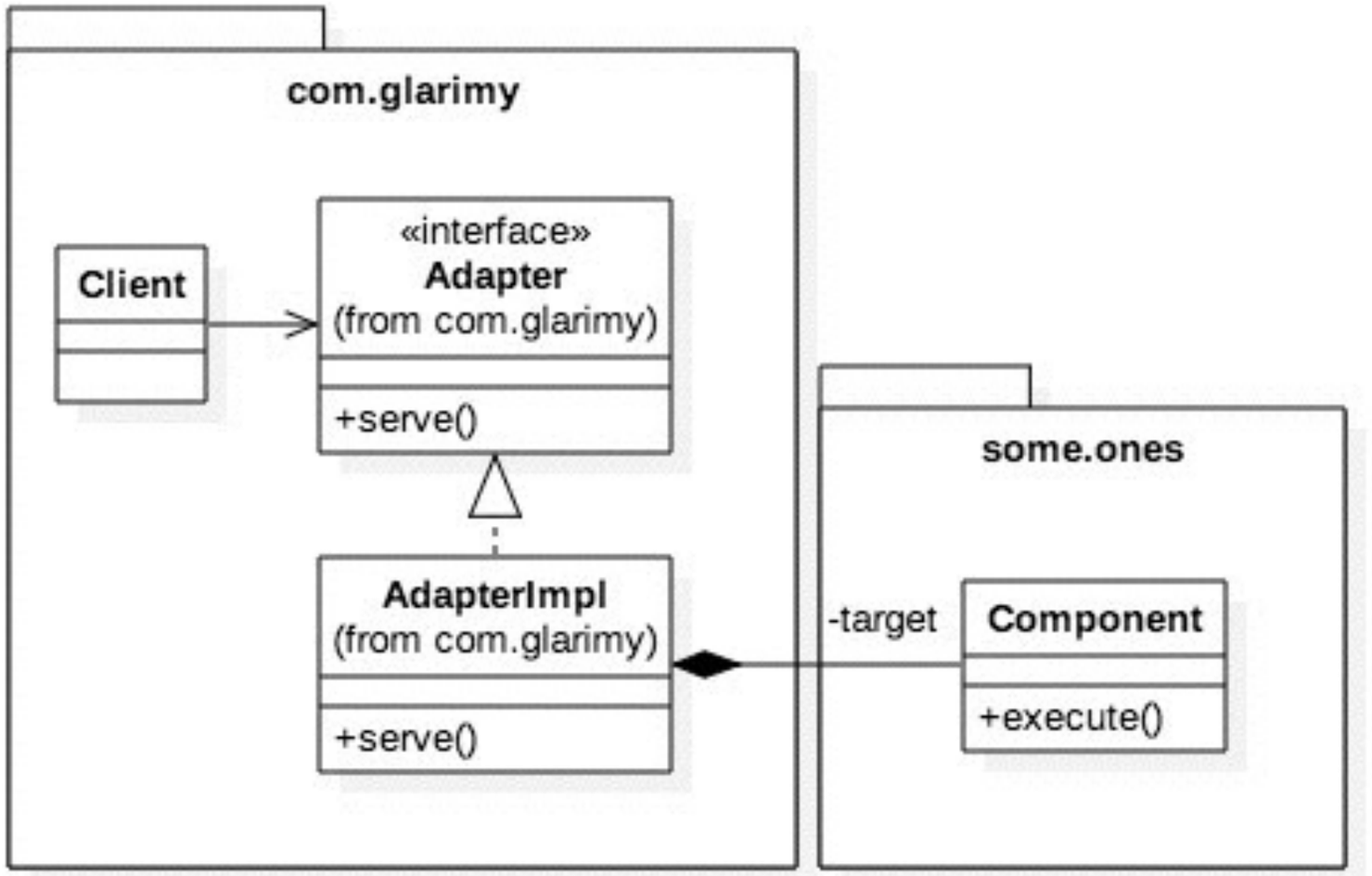


ADAPTER – PROBLEM ILLUSTRATION

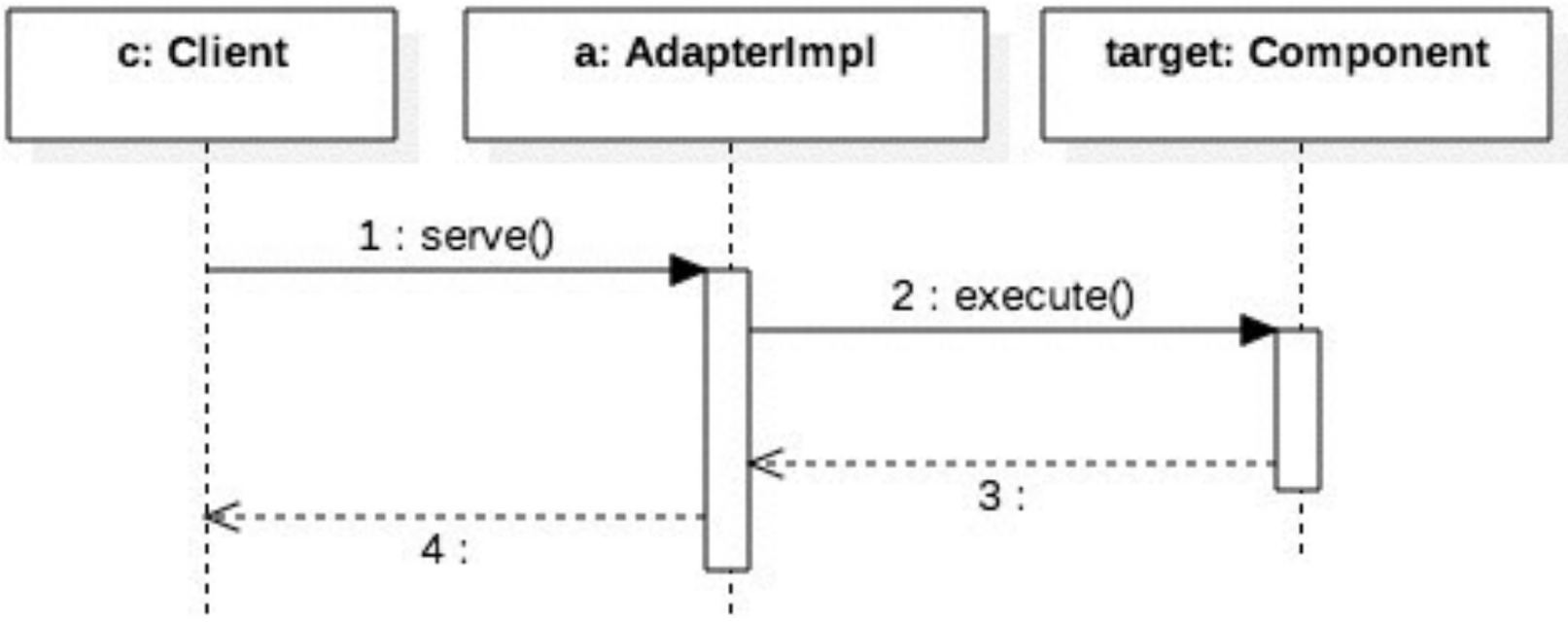
```
package some.ones;

public class Component {
   public void execute(){
      System.out.println("some.ones.Component::execute");
   }
}
```

```
import some.ones.Component;

public class Client {

   public static void main(String[] args) {
      Component component = new Component();
      component.execute();
   }
}
```

ADAPTER – SOLUTION



ADAPTER – SOLUTION



ADAPTER – SOLUTION

```
public interface Adapter {
   public void serve();
}
```

```
public class Client {

   public static void main(String[] args) {
      Adapter adapter = new AdapterImpl();
      adapter.serve();
   }
}
```

ADAPTER – SOLUTION

```
package some.ones;

public class Component {
    public void execute(){
        System.out.println("some.ones.Component::execute");
    }
}
```

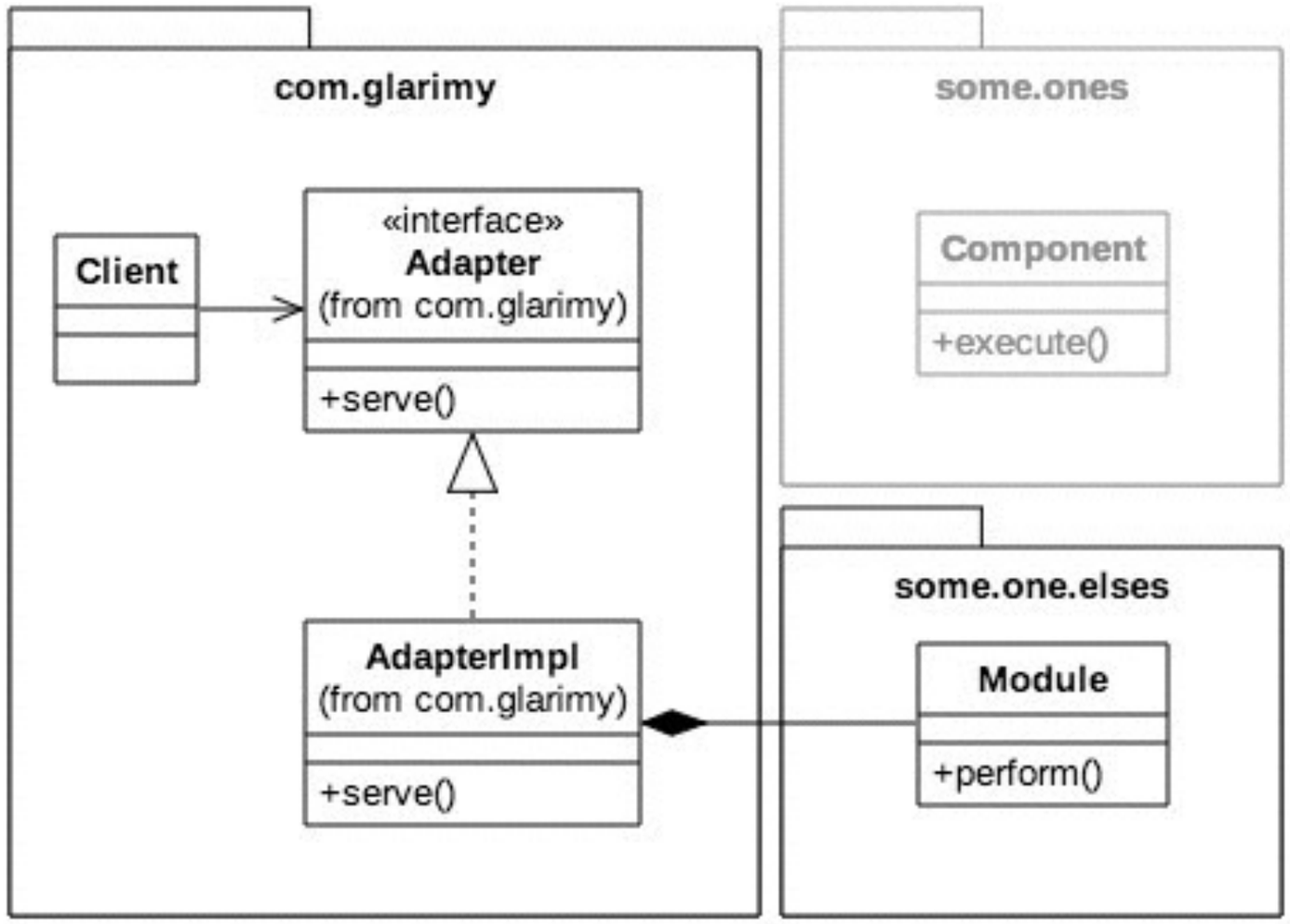
```
import some.ones.Component;

public class AdapterImpl implements Adapter {
    private Component component = new Component();

    @Override
    public void serve() {
        component.execute();

    }
}
```

ADAPTER – SOLUTION ILLUSTRATION

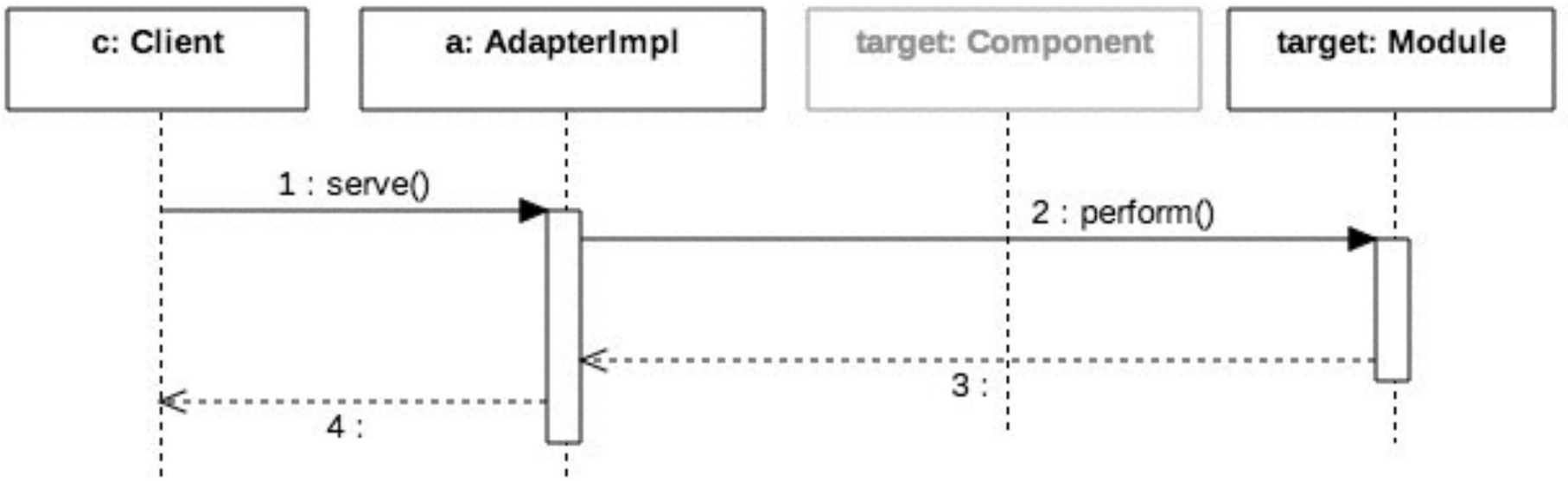
```
package some.one.elses;

public class Module {
    public void perform(){
        System.out.println("som.one.elses.Module::perform");
    }
}
```

```
import some.one.elses.Module;

public class AdapterImpl implements Adapter {
    private Module module = new Module();

    @Override
    public void serve() {
        module.perform();
    }
}
```

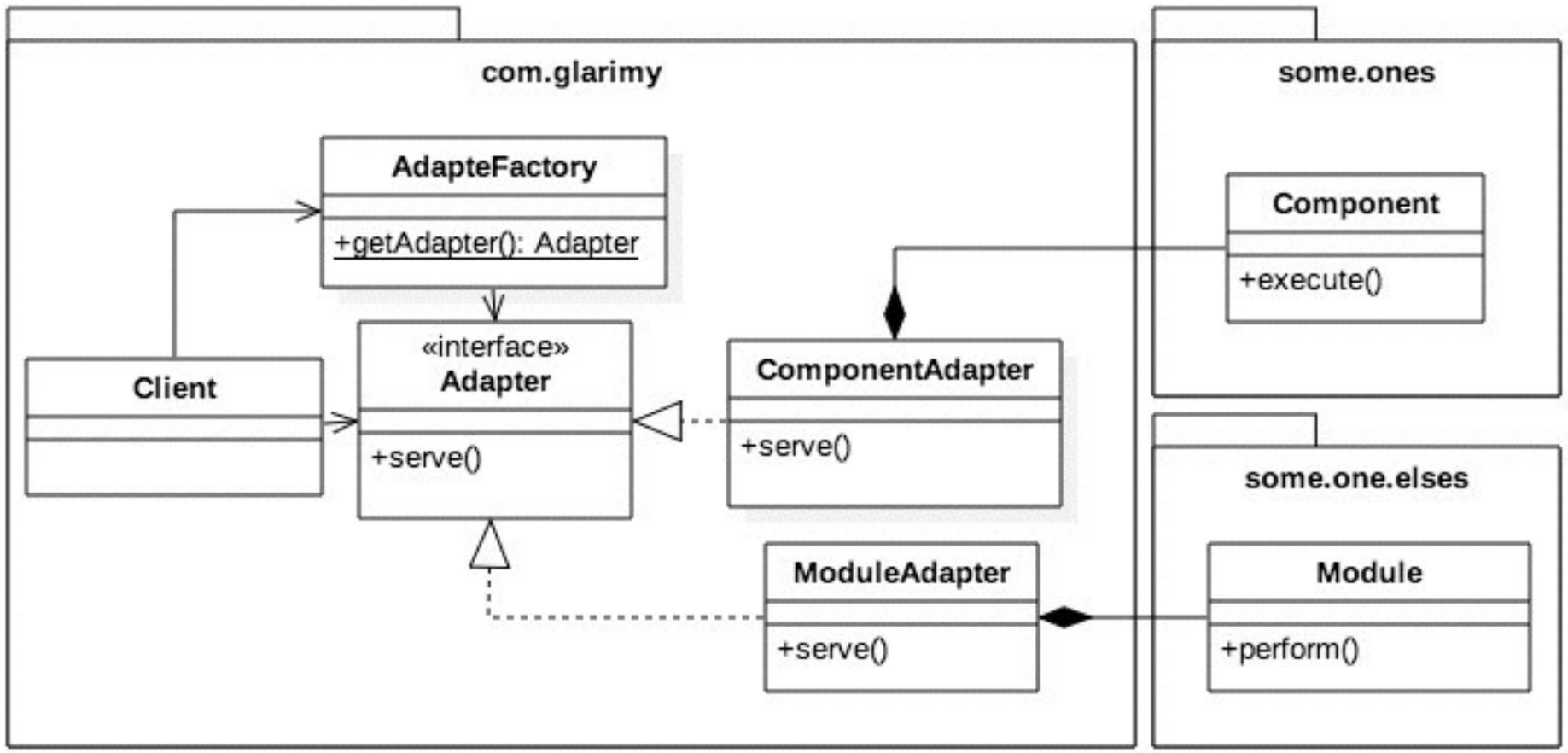
ADAPTER – SOLUTION ILLUSTRATION



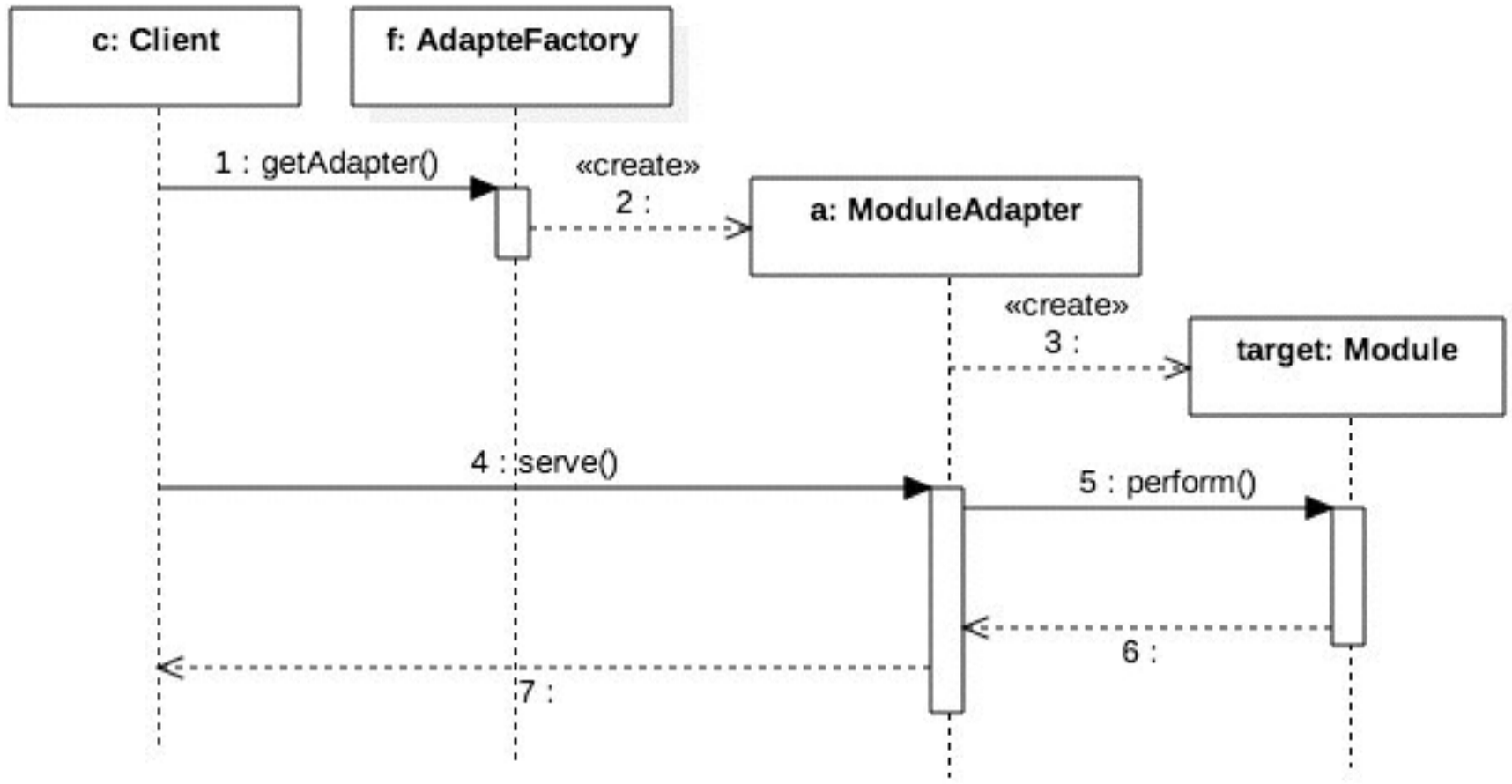
ADAPTER – SOLUTION ILLUSTRATION



ADAPTER – USING FACTORY



ADAPTER – USING FACTORY



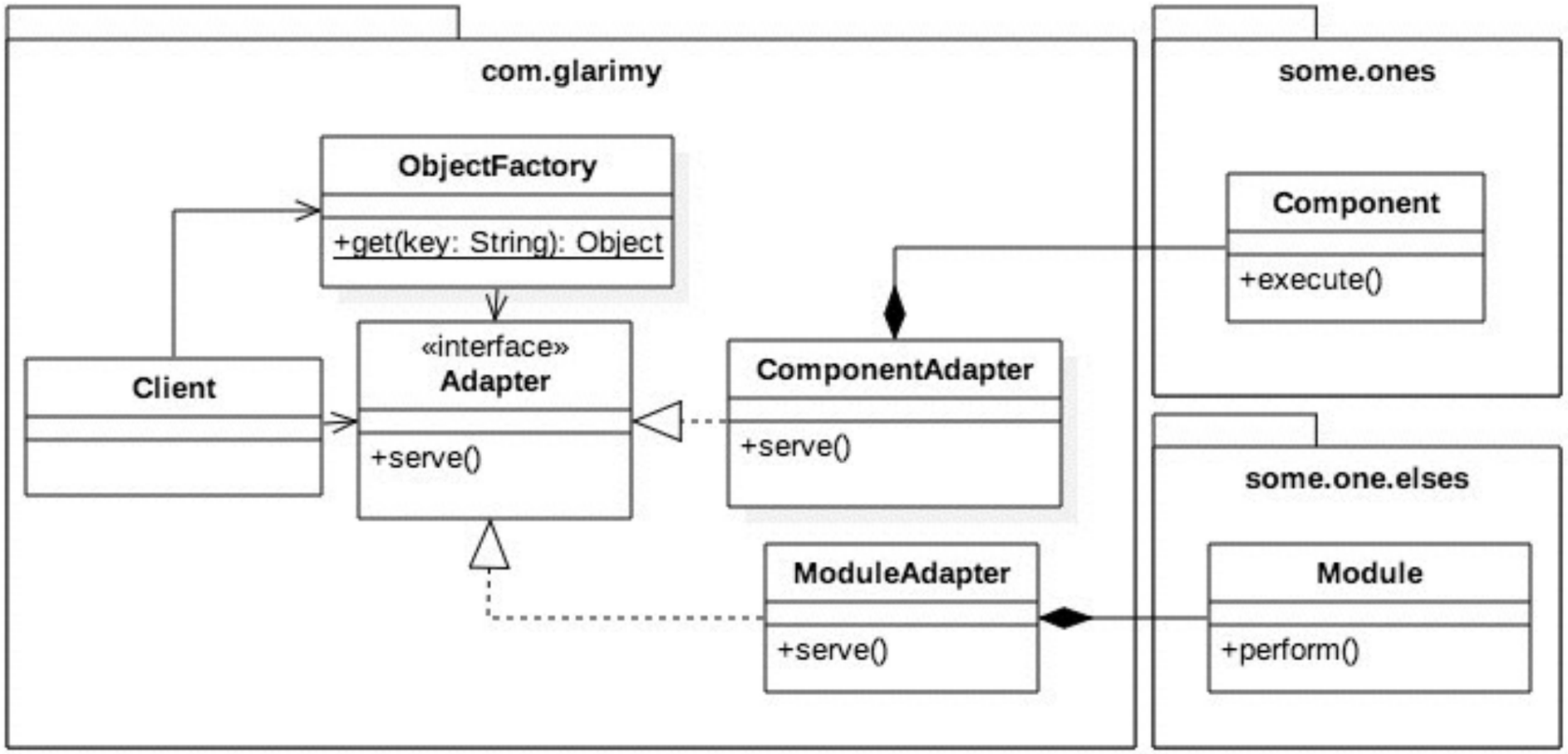
ADAPTER – USING FACTORY

```
public class AdapterFactory {
    public static Adapter getAdapter() throws Exception {
        Properties props = new Properties();
        props.load(new FileReader("adapter.properties"));
        String library = props.getProperty("library");
        if (library.equals("component"))
            return new ComponentAdapter();
        else if (library.equals("module"))
            return new ModuleAdapter();
        else
            throw new Exception("Adapter not found for the referenced library");
    }
}
```

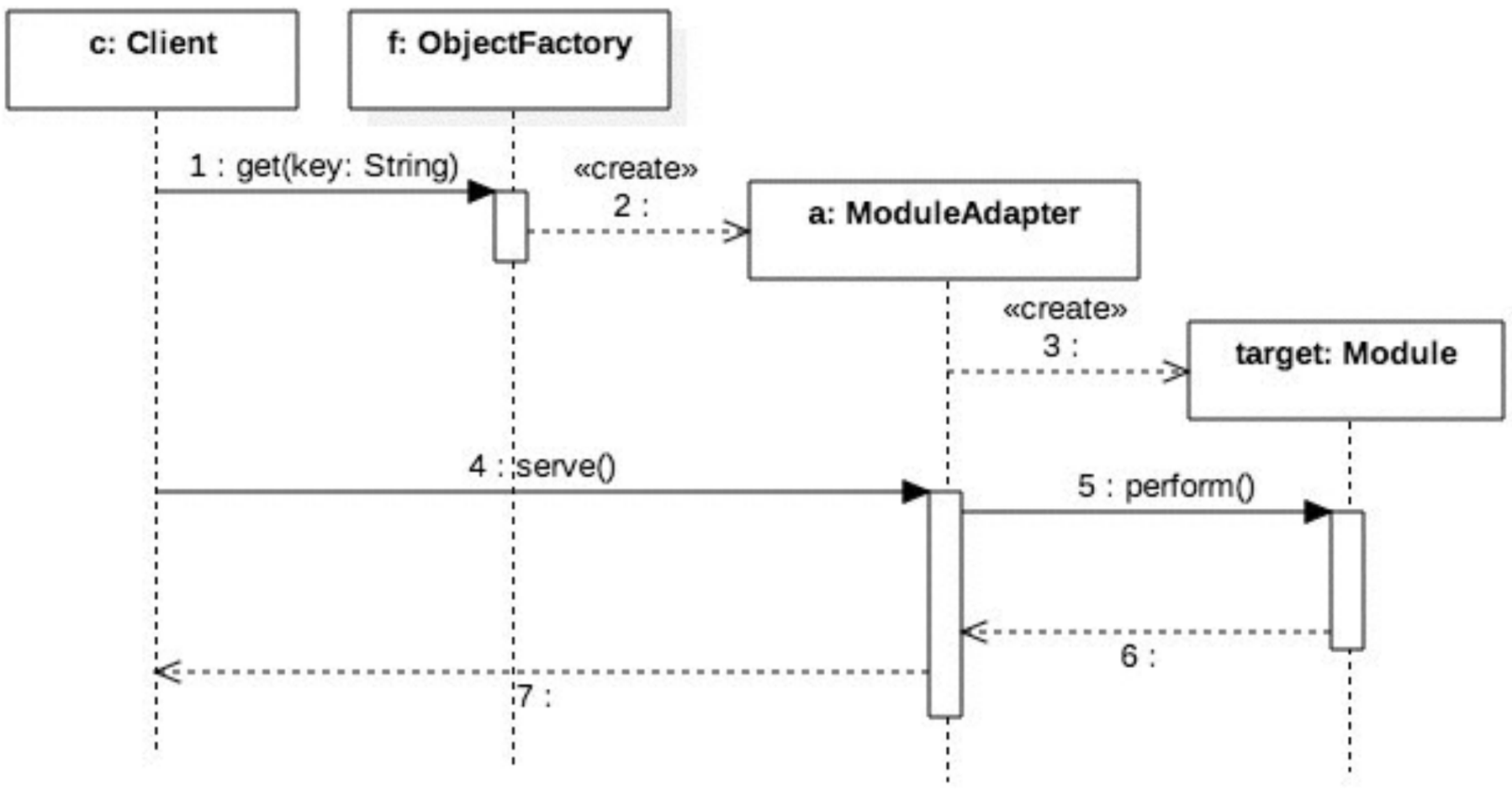
```
public class Client {

    public static void main(String[] args) throws Exception{
        Adapter adapter = AdapterFactory.getAdapter();
        adapter.serve();
    }
}
```

```
adapter.properties
library=component
```

ADAPTER – USING COMMON FACTORY



ADAPTER – USING COMMON FACTORY



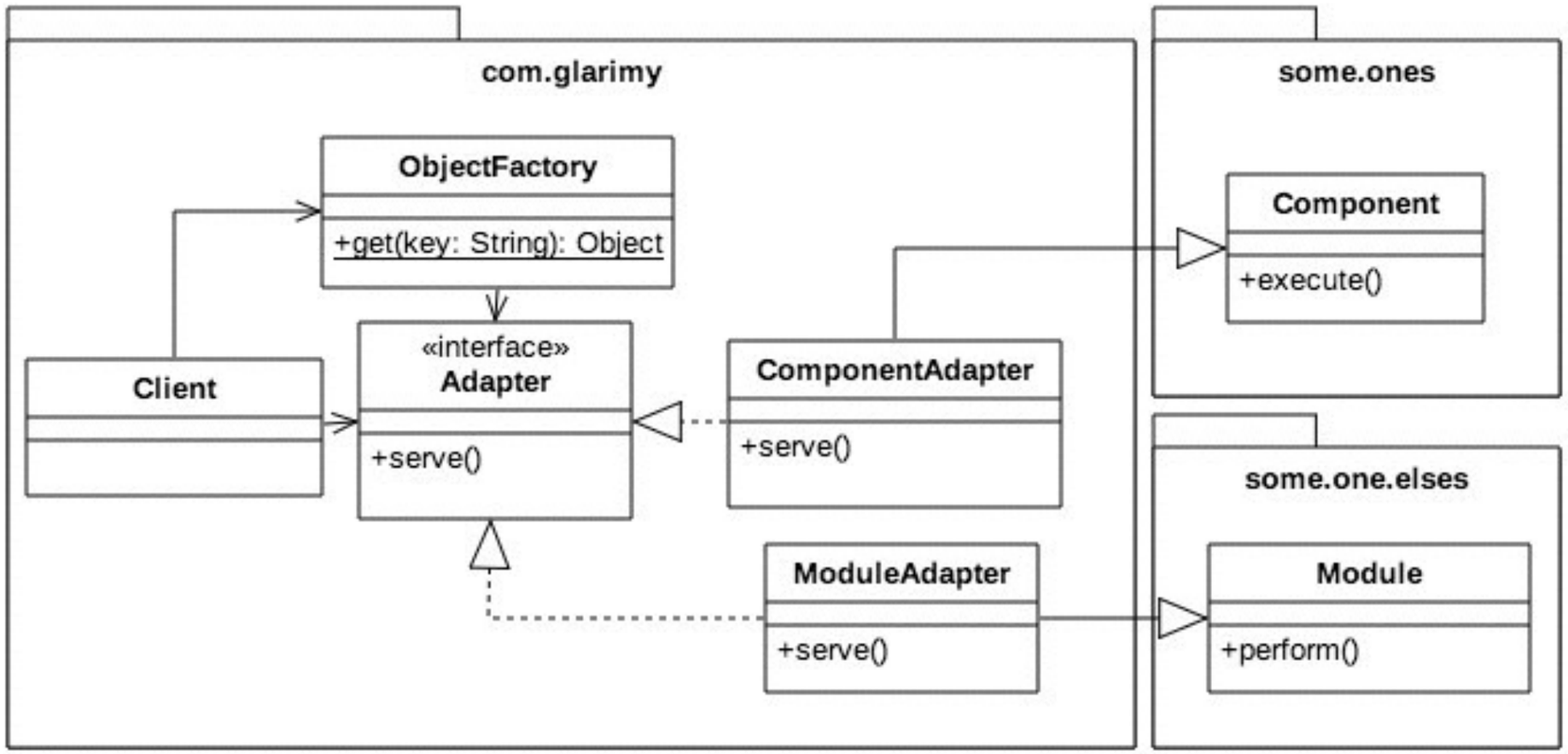
ADAPTER – USING COMMON FACTORY

```
public class ObjectFactory {
   public static Object get(String key) throws Exception {
      Properties props = new Properties();
      props.load(new FileReader("factory.properties"));
      String adapter = props.getProperty(key);
      return Class.forName(adapter).newInstance();
   }
}
```

```
public class Client {

   public static void main(String[] args) throws Exception {
      Adapter adapter = (Adapter) ObjectFactory.get("adapter");
      adapter.serve();
   }
}
```

```
factory.properties
adapter=com.glarimy.adapter.ModuleAdapter
```

ADAPTER – USING GENERALISATION

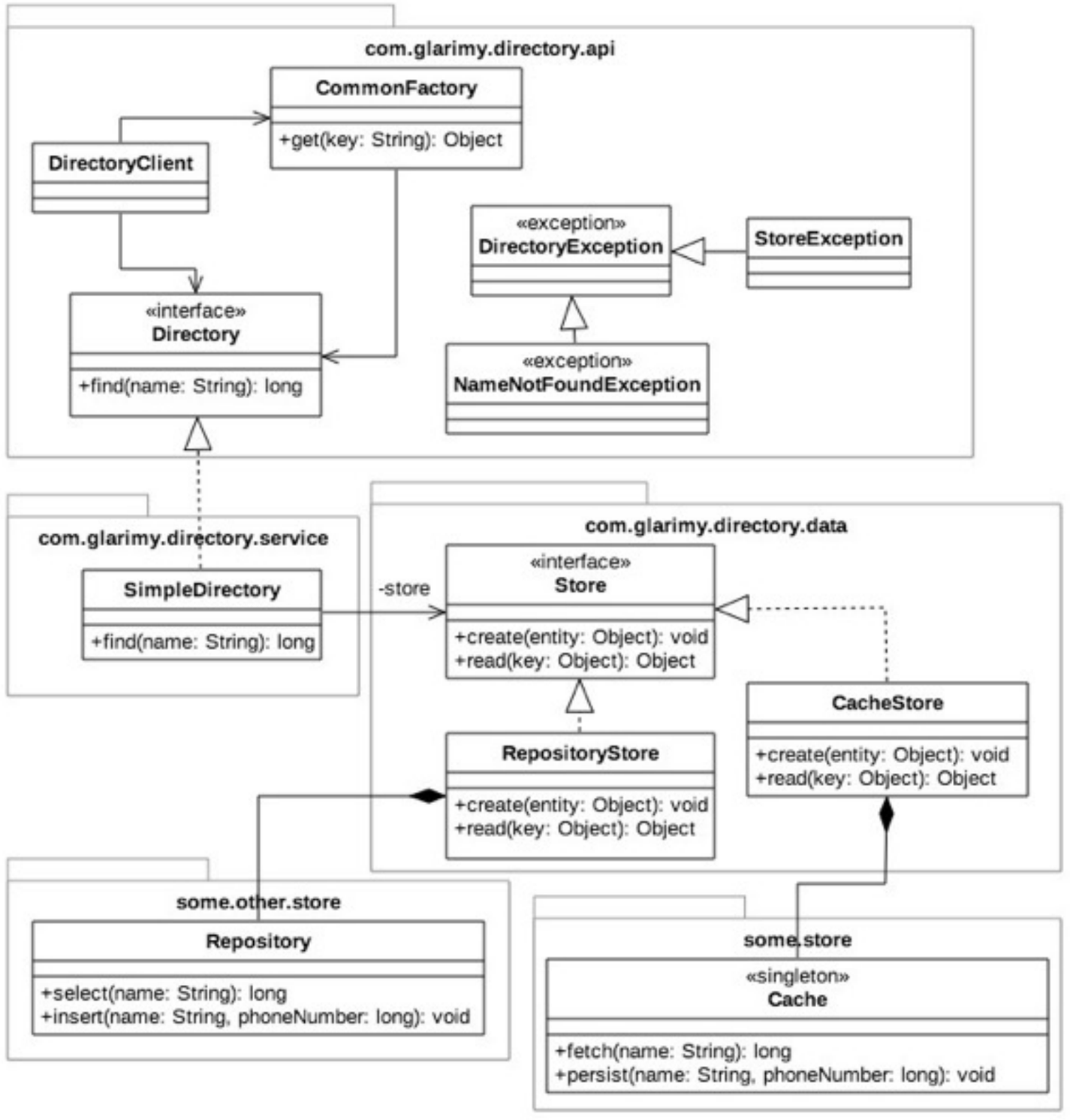


ADAPTER – USING GENERALISATION

```
import some.ones.Component;

public class ComponentAdapter extends Component implements Adapter {

    @Override
    public void serve() {
        super.operate();
    }
}
```

```
import some.one.elses.Module;

public class ModuleAdapter extends Module implements Adapter {

    @Override
    public void serve() {
        super.execute();
    }
}
```

ADAPTER – CASE STUDY



ADAPTER – CASE STUDY

```
package com.glarimy.directory.api;

public interface Directory {
    public long find(String name) throws NameNotFoundException, DirectoryException;
}
```

```
public class CommonFactory {
    public static Object get(String key) throws Exception {
        Properties props = new Properties();
        props.load(new FileReader("factory.properties"));
        String name = props.getProperty(key);
        @SuppressWarnings("rawtypes")
        Class claz = Class.forName(name);
        return claz.newInstance();
    }
}
```

```
public class DirectoryClient {
    public static void main(String[] args) throws Exception {
        Directory dir = (Directory) CommonFactory.get("directory");
        long phoneNumber = dir.find("Krishna");
        System.out.println(phoneNumber);
    }
}
```

ADAPTER – CASE STUDY

```
public interface Store {
    public void create(Object pk, Object entity) throws StoreException;

    public Object read(Object pk) throws StoreException;
}
```

```
public class SimpleDirectory implements Directory {
    private Store store = null;

    public SimpleDirectory() throws DirectoryException {
        try {
            store = (Store) CommonFactory.get("store");
            store.create("Krishna", 9731423166L);
        } catch (Exception e) {
            throw new DirectoryException();
        }
    }

    public long find(String name) throws DirectoryException {
        Long phoneNumber;
        try {
            phoneNumber = (Long) store.read(name);
            if (phoneNumber == null)
                throw new NameNotFoundException();
            return phoneNumber;
        } catch (StoreException e) {
            throw new DirectoryException();
        }
    }
}
```

ADAPTER – CASE STUDY

```
package some.store;

import java.util.HashMap;
import java.util.Map;

public class Cache {
    private Map<String, Long> entries;
    private static Cache instance;

    public static Cache getInstance() {
        if (instance == null)
            instance = new Cache();
        return instance;
    }

    private Cache() {
        entries = new HashMap<>();
        entries.put("Krishna", 9731423166L);
    }

    public long fetch(String name) {
        return entries.get(name);
    }

    public void persist(String name, long phoneNumber){
        entries.put(name, phoneNumber);
    }
}
```

ADAPTER – CASE STUDY

```
import some.store.Cache;

public class CacheStore implements Store {
    private Cache cache = Cache.getInstance();

    @Override
    public void create(Object pk, Object entity) throws StoreException {
        try {
            cache.persist((String) pk, (Long) entity);
        } catch (Exception e) {
            throw new StoreException(e);
        }
    }

    @Override
    public Object read(Object pk) throws StoreException {
        try {
            return cache.fetch((String) pk);
        } catch (Exception e) {
            throw new StoreException(e);
        }
    }

}
```

ADAPTER – CASE STUDY

```
package some.other.store;

public class Repository {
	private Connection connection;
	private PreparedStatement selectStatement;
	private PreparedStatement insertStatement;

	public Repository() throws Exception {
		Class.forName("com.mysql.jdbc.Driver");
		connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/glarimy?user=root&password=admin");
		selectStatement = connection.prepareStatement("select * from directory where name=?");
		insertStatement = connection.prepareStatement("insert into directory (name, phoneNumber) values (?, ?)");
	}

	public long select(String name) throws Exception {
		selectStatement.setString(1, name);
		ResultSet rs = null;
		try {
			rs = selectStatement.executeQuery();
			if (rs.next())
				return rs.getLong("phonenumber");
			else
				throw new Exception("No contact found!");
		} finally {
			rs.close();
		}
	}

	public void insert(String name, long phoneNumber) throws Exception {
		insertStatement.setString(1, name);
		insertStatement.setLong(2, phoneNumber);
		insertStatement.executeUpdate();
	}

	public void finalize() {
		. . .
	}
}
```

ADAPTER – CASE STUDY

```
package com.glarimy.directory.data;

public class RepositoryStore implements Store {
     private Repository repo;

     public RepositoryStore() throws StoreException {
          try {
               repo = new Repository();
               repo.insert("Krishna", 9945500066L);
          } catch (Exception e) {
               throw new StoreException(e);
          }
     }

     @Override
     public void create(Object pk, Object entity) throws StoreException {
          try {
               repo.insert((String) pk, (Long) entity);
          } catch (Exception e) {
               throw new StoreException(e);
          }
     }

     @Override
     public Object read(Object pk) throws StoreException {
          try {
               return repo.select((String) pk);
          } catch (Exception e) {
               throw new StoreException(e);
          }
     }
}
```

ADAPTER – RESOURCES

- GIT Source Code
  - /glarimy
- YouTube Channel
  - sversity-glarimy
- Website
  - www.glarimy.com
  - http://sversity.glarimy.com
- Facebook
  - /glarimy