# Naive container

- A very simple and naive dependency injection container

## Dependencies

- Dependencies are injected via constructors
- Each constructor parameter defines one dependency
- There are two types of dependencies
- Simple dependencies (`Dependency<T>` class) are characterized solely by a type (`Class<T>`)

  ```
  ctor(Service1 srv, String s, Integer i){...}
  ```

- Named dependencies (`NamedDependency<T>` class) are characterized by a type (`Class<T>`) and by a name (`String`)
  - The name is defined in parameter annotation

  ```
  ctor(@Named("host") String s, @Named("port") Integer i){...}
  ```

## Bindings

- A binding defines a mapping between a dependency and a way to obtain instances.
- There are three diferent binding types
  - `TypeBinding`, binds a dependency to a concrete type. Each binding usage will produce a new instance of that type.
  - `InstanceBinding`, binds a dependency to an instance
  - `SingletonBinding`, binds a dependency to a concrete type. Each bingind usage will return the *same* instance of that type.

## Injector

- An injector creates instances and injects dependencies into them

```
public interface Injector {
  public <T> T getInstanceOfExactType(Class<T> type);
  public <T> T getInstance(Dependency<T> type);
  public <T> T getInstance(Class<T> type);
  public void addBinding(Binding<?> b);
}
```

## Configuration

- An `InjectorConfiguration` configures an injector, i.e., adds a set of bindings to an injector.

```
public interface InjectorConfiguration {
        public void configure(Injector injector) throws NaiveContainerException;
}
```