

# Quiz 3: Lecture 1 Review Questions

**Question:** How can you determine if a class adheres to the SRP?

- Attempt to describe the class in more than once sentence, using the words "if" and "then"
- Attempt to describe the class in once sentence, using the words "and" and "or"
- Attempt to describe the class in once sentence, without using the words "and" and "or"
- Attempt to describe each of the class' methods in once sentence

**Question:** What's the difference between loosely coupled objects and tightly coupled objects?

- Loosely coupled objects behave like a single unit, tightly coupled objects are more flexible and reusable
- Tightly coupled objects are easy to understand and maintain, loosely coupled objects are like spaghetti code
- Tightly coupled objects are easy to implement and debug, loosely coupled objects are expensive to develop
- Loosely coupled objects have very few dependencies between components, tightly coupled object have many dependencies

**Question: What are FOUR benefits that stepwise refinement offers? (Please alphabetize answer ordering by the first letter, do not use indentation)**

Drag from here

Construct your solution here

**Achieve** incremental development on a very fine level of granularity

**Produce** working (**and** tested) prototypes of software **as** it develops

**Perform** iterative development using coarsely-grained detail

**Concentrate** on the most relevant "chunks"  
at the current phase of development

**Ignore** unrelated details

Reset

---

Submit test

## Quiz 3: Lecture 1 Review (/tests/3) **Solutions**

---

Marks Obtained : 3

### Question 1

How can you determine if a class adheres to the SRP?

**Option 1:** Attempt to describe the class in more than once sentence, using the words "if" and "then"

**Option 2:** Attempt to describe the class in once sentence, using the words "and" and "or"

**Option 3:** Attempt to describe the class in once sentence, without using the words "and" and "or"

**Option 4:** Attempt to describe each of the class' methods in once sentence

**Your Answer:** Attempt to describe the class in once sentence, without using the words "and" and "or"

## Correct!

### Question 2

What's the difference between loosely coupled objects and tightly coupled objects?

**Option 1:** Loosely coupled objects behave like a single unit, tightly coupled objects are more flexible and reusable

**Option 2:** Tightly coupled objects are easy to understand and maintain, loosely coupled objects are like spaghetti code

**Option 3:** Tightly coupled objects are easy to implement and debug, loosely coupled objects are expensive to develop

**Option 4:** Loosely coupled objects have very few dependencies between components, tightly coupled object have many dependencies

**Your Answer:** Loosely coupled objects have very few dependencies between components, tightly coupled object have many dependencies

# Correct!

**Question:** What are FOUR benefits that stepwise refinement offers? (Please alphabetize answer ordering by the first letter, do not use indentation)

Drag from here

**Perform** iterative development using coarsely-grained detail

Construct your solution here

**Achieve** incremental development on a very fine level of granularity

**Concentrate** on the most relevant "chunks" at the current phase of development

**Ignore** unrelated details

**Produce** working (**and** tested) prototypes of software **as** it develops

Reset

Get feedback

# Correct!

## Quiz 4: Coupling and Cohesion **Question**

**Question:** Use the principles of coupling and cohesion to create a new version of the previous ruby Dictionary code that is more loosely coupled and more highly cohesive.

Drag from here

```
class Dictionary
```

```
class TextDictionary < Dictionary
```

```
def write
```

```
  # write XML to @file using the @definitions hash
```

```
end
```

```
def read
```

```
  # read XML from @file and populate the @definitions hash
```

```
end
```

```
end
```

```
end
```

```
end
```

```
def initialize(file)
```

```
  @definitions = Hash.new
```

```
  @file = file
```

```
end
```

```
def self.instance(file)
```

```
  if File.extname(file) == ".xml"
```

```
    XMLDictionary.new(file)
```

```
  else
```

```
    TextDictionary.new(file)
  end
end
```

```
def add_definition(term, definition)
  @definitions[term] = definition
end
```

```
class XMLDictionary < Dictionary
```

```
def write
  # write text to @file using the @definitions hash
end
def read
  # read text from @file and populate the @definitions hash
end
```

```
dictionary = Dictionary.instance("dictionary.txt")
dictionary.add_definition("autodidact",
  "someone who learned without a teacher")
dictionary.add_definition("cogent",
  "clear, logical, and convincing")
dictionary.add_definition("pedagogy",
  "the method and practice of teaching")
dictionary.write
dictionary.read
```

Construct your solution here

Reset

Submit test

## Quiz 4: Coupling and Cohesion (/tests/4) **Solution**

Marks Obtained : 1

**Question:** Use the principles of coupling and cohesion to create a new version of the previous ruby Dictionary code that is more loosely coupled and more highly cohesive.

Drag from here

Construct your solution here

```
class Dictionary
```

```
  def initialize(file)
    @definitions = Hash.new
    @file = file
  end
```

```
  def add_definition(term, definition)
    @definitions[term] = definition
  end
```

```
  def self.instance(file)
    if File.extname(file) == ".xml"
      XMLDictionary.new(file)
    else
      TextDictionary.new(file)
    end
  end
```

```
end
```

```
class XMLDictionary < Dictionary
```

```
def write
  # write XML to @file using the @definitions hash
end
def read
  # read XML from @file and populate the @definitions hash
end
```

end

```
class TextDictionary < Dictionary
```

```
  def write
    # write text to @file using the @definitions hash
  end
  def read
    # read text from @file and populate the @definitions hash
  end
```

end

```
dictionary = Dictionary.instance("dictionary.txt")
dictionary.add_definition("autodidact",
  "someone who learned without a teacher")
dictionary.add_definition("cogent",
  "clear, logical, and convincing")
dictionary.add_definition("pedagogy",
  "the method and practice of teaching")
dictionary.write
dictionary.read
```

Reset

Get feedback

# Correct!





# Quiz 5: SOLID Principles Questions

**Question:** Which of the SOLID principles can be best used here to add the new broadcasting platforms in a well-designed manner?

- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

**Question:** Add in functionality for the **Newsperson** class to be able to make broadcasts using either the **Twitter**, **TV**, or **Newspaper** news broadcasting platforms. (Please use the order of **Newsperson** - **Newspaper** - **Twitter** - **Television** for class ordering.)

Drag from here

Construct your solution here

end

end

```
def broadcast(news)
  tweets news
end
```

```
class Newspaper
```

```
class Television
```

```
class Twitter
```

end

```
laura = Newsperson.new
laura.broadcast("Breaking news!")
laura.broadcast("Breaking news!",
Twitter)
```

```
class Newsperson
```

```
end
```

```
def broadcast(news, platform = Newspaper)
  platform.new.broadcast(news)
end
```

```
def broadcast(news)
  live_coverage news
end
```

```
def broadcast(news)
  prints news
end
```

Reset

Submit test

## Quiz 5: SOLID Principles (/tests/5) **Solutions**

---

Marks Obtained : 2

### Question 1

Which of the SOLID principles can be best used here to add the new broadcasting platforms in a well-designed manner?

- Option 1:** Open/Closed Principle
- Option 2:** Liskov Substitution Principle
- Option 3:** Interface Segregation Principle
- Option 4:** Dependency Inversion Principle

**Your Answer:** Dependency Inversion Principle

## Correct!

**Question:** Add in functionality for the Newsperson class to be able to make broadcasts using either the Twitter, TV, or Newspaper news broadcasting platforms. (Please use the order of Newsperson - Newspaper - Twitter - Television for class ordering.)

Drag from here Construct your solution here

```
class Newsperson
```

```
  def broadcast(news, platform = Newspaper)  
    platform.new.broadcast(news)  
  end
```

```
end
```

```
class Newspaper
```

```
def broadcast(news)
  prints news
end
```

```
end
```

```
class Twitter
```

```
def broadcast(news)
  tweets news
end
```

```
end
```

```
class Television
```

```
def broadcast(news)
  live_coverage news
end
```

```
end
```

```
laura = Newsperson.new
laura.broadcast("Breaking news!")
laura.broadcast("Breaking news!",
Twitter)
```

Reset

Get feedback

# Correct!