CSU Sacramento, Spring 2019 **Dr.Jagan Chidella**

CSC 131-02,03,05 – End Exam SET-A KEY

Student Name:_________________________

Section: ____

PLEASE WRITE YOUR ANSWERS LEGIBLY

A. TRUE or FALSE - (10*1 = 10 points)

1. Design Patterns describe how to structure classes to meet a given requirement. True
2. Strongly coupled modules are easy to change. False
3. REST is an architectural style protocol. True
4. HTTP Status Code, 5xx is a client error. False
5. Scrum master is ultimately responsible for quality of product outcome. False
6. A good software has high coupled and less cohesive modules. False
7. Synchronization is difficult in Event-Driven architecture. True
8. In MVC architecture, Controller is a request handler. True
9. Decorator pattern enhances capability of individual object. True
10. Use case diagram is a dynamic/behavior diagram which models system functionality using actors & use cases. True

B. Match using ALPHABET only (Choose the best answer) (10*1=10 points)

1. Apache Tomcat	C	a) Holds meta information
2. Design Pattern	J	b) Agile project management tool
3. HTTP Status code	I	c) Webserver
4. Flying Donut	B	d) Independent paths in program.
5. Standard Formats	F	e) Rest Client tool
6. GitHub/GitLab	H	f) JSON and XML
7. REST HTTP header(s)	A	g) Project management tool for handling dependencies
8. Cyclomatic Complexity	D	h) Version Control service
9. Apache Maven	G	i) Predefined status of task done at server.
10. Postman	E	j) Reusable solution

C. Fill in blanks (Total 10 fill in blanks). (10*2 = 20 points)

1. REST webservices use __**HTTP**_____ protocol as a communication medium between client & server.
2. Composition is ___**HAS-A**____ relationship and Inheritance is _____**IS-A**_______ relationship. (1 point each)
3. High cohesion tends to reduce ____**COUPLING**_____ between modules.
4. Inheritance has higher amount of _____**COUPLING**______ compared to Composition.
5. The _____**STRATEGY**________ design pattern (also known as policy pattern) is a behavioral software design pattern that enables selecting an algorithm at runtime.
6. REST is an example for __________**CLIENT-SERVER**__________ architecture.
7. Redirecting results/output of an operation into a file instead of output screen is an example for _________**PIPE AND FILTER**________ architecture.
8. Generators and receivers communicate through an intermediary (called an event queue). This is _______**EVENT-DRIVEN**_______ architecture.
9. The ___**ITERATOR**_________ pattern provides a way to access the elements of an aggregate object without exposing its underlying representation.
10. In ____**FACTORY**_________ design pattern, we create object without exposing the creation logic to client and the client use the same common interface to create new type of object.

CSU Sacramento, Spring 2019 **Dr.Jagan Chidella**

CSC 131-02,03,05 – End Exam SET-A KEY

Student Name:________________________

Section: ____

D. Provide LEGIBLE answers for the following questions. (15*2=30 points)

1. Define Code Refactoring and explain its purpose? (1+1)

   The process of changing code without altering its functional behavior.

   Purpose: To improve code for readability/performance/security/compatibility etc.

2. List at least 3 components of a REST Request call?

   1. URL, 2. METHOD(GET,POST,PUT,DELETE), 3. HEADERS, 4. BODY (DATA)

3. Define Burndown chart and explain its purpose? (1+1)

   It is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal. That is, it is a run chart of outstanding work. **Purpose:** It is useful for predicting when, the work will be completed.

4. Define stateless protocol? Provide an example. (1+1)
   A stateless protocol does not require the server to retain session information or status about each communicating partner for the duration of multiple requests. Every request is independent of previos request. **Example:** HTTP

5. a) What is the status code returned if POST call is successful? ____**201**________

   b) Name the header you used to perform Content Negotiation? ___**Accept/ContentType**_____

6. Explain different types of Restful URIs?
   1. Instance Based 2. Collection Based

7. Explain Halstead Complexity?

   Halstead Complexity (calculated from the number of **operators** and **operands**).

   It reflects the implementation or expression of algorithms in different languages, but be independent of their execution on a specific platform

8. Explain Encapsulation/Information Hiding? What is its purpose?

   Wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Variables are made private and methods are made public.

   Purpose: To hide sensitive data/information.

9. Transform the following XML into JSON response

<City>
<id> 1231 </id>
<name> Oakland </name>
</City>
	➔	{
		"id": "1231",
		"name": "Oakland"
		}

10. What is supported as a part of Level 3 HATEOAS?

Support Responses which contains that a Client can use. Response contains link to access the same resource itself.

11. Identify the type of coupling in following piece of code and provide justification? (1+1)

String item = “peanut”;
void main(){
System.out.println(item);
flyDumbo();
System.out.println(item);
}
private void flyDumbo(){
item = ”feather”;
}
	➔	Common Coupling.
		Justification: Both the modules use global variables.

12. List at least two advantages of layered architecture? (1+1)
    a) Improved cohesion which allows for multiple physical deployment options (e.g., presentation and application logic on one machine; storage on another)
    b) Decreased coupling which promotes re-use
    c) Specialized expertise
    d) Multiple applicants can reuse the components.
    e) We can test the components independently.
    f) Increases flexibility, scalability, maintainability.

13. List one advantage and one disadvantage with Client-Server architecture? (1+1)

    Advantage:
    a) Centralized Control
    b) Improves Data sharing.

    Disadvantage:
    a) "Single" point of failure
    b) Difficult to scale.

14. Imagine you are developing a weather system, where change in weather must be reflected in Views to show to public. All weather changes in system must be notified to all corresponding views.
    Can you identify this design pattern?
    Observer/Listener Pattern

Student Name:_________________________

Section: ____

15. Describe
    a) Product Backlog grooming:
    Backlog grooming ensures that backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery.

    b) Sprint Backlog Selection Process:
    Sprint backlog is a task list identified by Scrum team to be completed during Scrum sprint. During sprint planning meeting, the team selects some number of product backlog items (based on priority), usually in form of user stories & identifies tasks necessary to complete each user story.

E. Provide Legible answers for following questions (4*5=20)

1. Github work cycle involves following actions (Commit, Modify, Resolve, Pull, Push). Order these actions based on when they would be performed in normal work cycle. You may use an action more than once.

1. Pull 2. Resolve* 3. Modify 4. Commit 5.Pull 6. Resolve* 7. Push

2. Define Coupling and different types of coupling?

Coupling: Coupling between two modules is a measure of the degree of interdependence or interaction between the two modules.

Types:

Data coupling: Modules communicate through a parameter.

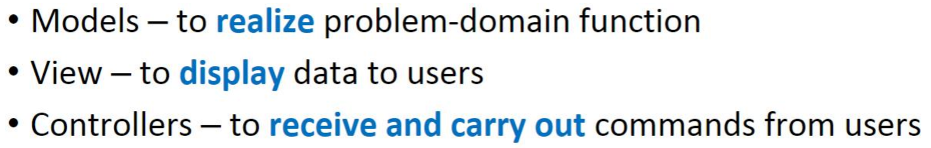
Stamp coupling: Modules communicate using a composite data item.

Control coupling: Control coupling exists between two modules, if data from one module is used to direct the order of instructions execution in another.

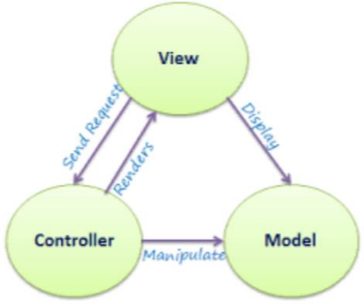
Common coupling: Two modules are common coupled, if they share data through some global data items.

Content coupling: Content coupling exists between two modules, if they share code, e.g. a branch from one module into another module.

3. Explain MVC architecture using the help of a diagram?



- Models – to **realize** problem-domain function
- View – to **display** data to users
- Controllers – to **receive and carry out** commands from users



4. Explain how you may use any Design Pattern for your team project? Provide justification?
   You can possibly use the following Design Patterns
   1. Singleton: Implement this pattern at Server level, to make sure that all the clients/devices get the same instance when they request for an instance.
   2. Observer: Implement this pattern at Server level. Notifications from server are sent to all the clients.
   3. Iterator: Implement this pattern to access lists. Example: Iterate on list of lost devices, found devices etc.
   4. Composite Pattern: A list can hold list of devices. List of lost devices/found devices etc. List exhibits HAS-A relation.

F. Refactor the following code for encapsulation. (5 points)

public class School
{
public String name;
pubic String id;
 	
String getName(){
return name;
}
 	
String getId(){
return id;
}
}
	➔	public class School
		{
		private String name;
		private String id;
		
		public String getName(){
		return name;
		}
		
		public String getId(){
		return id;
		}
		} // Make sure variables are private

G. Refactor the following code to reduce coupling. Do not change the functional behavior (4). Explain the type of coupling present in your refactored code (1 points) (4+1=5 points)

String item = “peanut”;
 	
void main()
{
flyDumbo();
}
 	
private static void flyDumbo(){
print(item);
}
	➔	void main()
		{
		String item = “peanut”;
		flyDumbo(item);
		}
		
		private static void flyDumbo(String item){
		print(item);
		}

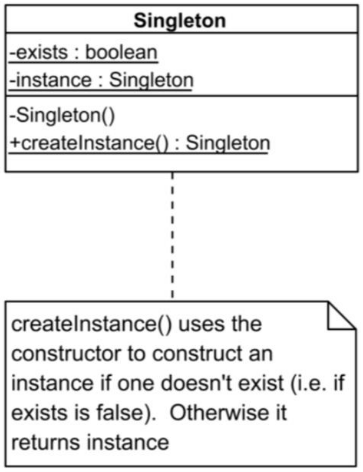
Coupling Type: COMMON		Coupling Type present in your code is: _DATA COUPLING__

H. Explain Singleton Design Pattern? Draw the UML diagram? List few applications? (3+4+3 = 10 points)

Singleton Pattern: (3 points)

Singleton pattern restricts the instantiation of a class to one "single" instance.
Constructor is made private to restrict instantiation from outside.

UML Diagram: (4 points)


Singleton
-exists : boolean
-instance : Singleton
-Singleton()
+createInstance() : Singleton
createInstance() uses the constructor to construct an instance if one doesn't exist (i.e. if exists is false). Otherwise it returns instance

Applications: (2 applications are enough to get 3 points)

1. Windowing systems with one event queue
2. Word processors with one menu bar (for all documents)
3. Hardware Interface access
4. Caching
5. Logger