Singleton Pattern:

Not more than one instance per JVM.

Class A

{

Private static A obj;

Private a()

{

}

Public static A getInstance()

{

Obj = new A();

Return obj;

}

}

Cloning – that can eliminate singleton .

Prototype:

We have an instance and we want to create new objects by copying that prototype.

1. Interfaces
2. Abstract Class

Public abstract class Bank

{

Public abstract Bank getName();

}

Public CitiBank extends Bank

{

@override

Public Bank getName()

{

Bank citi = new Bank();

Citi.name = “CitiBank”

Citi.ir = “”;

Return citi;

}

}

Factory

Abstract factory

Builder

Observer

Strategy

DI

IOC

3

5

3 5

If(number %3 === 0 && number % 5 == 0)

{

Syso(catdog)

}

Else if(number %3 ==0)

Cat

Else if(number % 5 ==0)

Dog

Else

Neither

Generics:

Collections:

List<Integer> myList = new ArrayList();

myList.add(1);

//myList.add(“Java”);

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| Dynamic Array | Doubly Linked List |
|  |  |
| Storing and accessing | Manipulation |

Map internally working.

String reverse.

JavaTPoint:

Spring:

DI:

IOC:

Framework used for J2EE.

Wrapping data and methods that operate on it in a single entity.

1. Modular
2. POJO -Plain Old Java Object.

1.public- anywhere

2. private- only with in class

3. Default- only in the package.

4. Protected: in the package as well as to the subclass outside the package.

1. Testability:
2. Lightweight:EJB
3. MVC,Transaction Management,
4. Central Exception Handling

Ioc Containers: create the objects,wiring them together, configuration and managing them

Entirely.

XML,3 onwards annotation,java

1. BeanFactory
2. ApplicationContext

**Bean: Object which is mainained by IOC containers.**

**Bean Lifecycle:**

**@Bean**

Bean Scopes:

1. Singleton- Default- one Instance per IOC @Singleton
2. Prototype@prototype
3. Request@RequestScope
4. Session @SessionScope
5. Global Session-@ApplicationScope
6. WebSocket

Internal Workings Of Map:

Hashing Principle.

Hashcode ,equals

Entry-Object

Bucket -Linked List ( Binary Search Tree- Java8 ).

Hash Collision

public class Reversal {  
  
  public static void main(String[] args) {  
         String str;  
         System.out.println("Enter your username: ");

// Java  
         Scanner scanner = new Scanner(System.in);  
         str = scanner.nextLine();  
         Reversal reverse = new Reversal();  
         String reversed = reverse.reverseString(str);  
         System.out.println("The reversed string is: " + reversed);  
     }  
  
     public  String reverseString(String str)  
     {  
         if (str.isEmpty())  
             return str;  
         return reverseString(str.substring(0, str.length()-1));  
        // return reverseString(str.substring(1)) + str.charAt(0);  
     }  
 }

[Reply in private](res://\\G2MResource_en.dll/%3cA%20HREF=%22%3conLeftClick%3eeCMD_SetChatTo%20103%3c/onLeftClick%3e%3conRightClick%3eeCMD_DoAttendeeContextMenu%206750220%3c/onRightClick%3e%22%3e%3c/A%3e)

**8:57 AM**

[**VISHAL PANNALA**](res://\\G2MResource_en.dll/%3cA%20HREF=%22%3conLeftClick%3eeCMD_SetChatTo%20105%3c/onLeftClick%3e%3conRightClick%3eeCMD_DoAttendeeContextMenu%206881292%3c/onRightClick%3e%22%3e%3c/A%3e)

public  String stringReverse(String str) {  
  if(str.length()== 0) {  
   return "";  
  }  
  else if(str.length() == 1) {  
   return ""+str.charAt(0);  
  }else {  
   return str.charAt(str.length()-1) + stringReverse(str.substring(0, str.length() -1));  
  }  
 }

Spring MVC:

MVC:

Separation Of Concern(SOC)

Model: Data layer

View : Presentation Layer

Controller: Business Logic,

IDE,

ANT

Maven

Pom,

Settings.xml

.m2

Source Code : GitHub

Gradle

Spring MVC Request flow:

FrontController/DispatcherServlet

HandlerMapping

Controller

Modelandview

Java is awesome

Avaj si emosewa

Java 8 Features: Stream, Functional Interfaces, Lambdas,

DatandTime,,Concurrency

PermGen removal

CRUD:

Create -Post

Read -Get

Update -Put

Delete-Delete

Http Verbs:

Stateless:

Resource:

|  |  |
| --- | --- |
| Retrieve info from server | Create a resource |
| They are always part of URI | As body |
| Less Secure | Secure |
| Restriction Data length | No such restrcition |
| Bookmarked/cached | Not Possible |

Idempotent:

Put /user/123

Post /user/123

Head/Patch/Options.

User

Id name email address

202 vishal [vishal@gmail.com](mailto:vishal@gmail.com) asdasdsdv

201 vishal

Rest vs

|  |  |
| --- | --- |
| Rest is an architecture | Protocol |
| No restriction on Data | XML |
| JSojn-Lightweighted and better performance | Heavy /not as good as rest |
| No Inherit security layer | Ws-Security |
| Http | TCP/Http |

Soap

Spring MVC:

Client- request-DispactherServlet-HandlerMappings- Controller-Return Model and View-DispatcherServlet-Send the Model Object to ViewResolver

DAO: data access Objects

One To Many

Many To Many:

404

400

200

201

304

500

502

@Controller @RestController

@Controller and @ResponseBody

@Qualifier

Interface Base

{

}

@Component(“sampleBase”)

Class SampleBase implements Base{}

@Component(“exampleBase)

Class ExampleBase implese Base{}

Class BaseController

{

@Autowired

@Qualifier(“exampleBAse”)

Base obj;

}

One minute please

@Configuration

@Component vs @Service Vs @Repository vs @ Configuration

Model- Data

ModelandView – data and UI

Sprint Cycle: 3 weeks

Team Size: 8-14

Qa,Devs,BA,Scrum Master, Devops

Overloading: happens in same class but either number of parameters or type of parameter must have to change.

Overriding: Happens when there is inheritance,parent-child relationship. 2 classes having same method signature but can have their own implementation.

Void sum (Int a , int b){  
}

Void sum(int a, double b){}

Void sum(int a,int b,int c)

Class A{

void sum(int a,int b)

{

print(a+b);

}

}

Class B extends A

{

Void sum(int a,int b)

{

Print(a+b+10)

}

}

Give me a minute please

@Transactional:

SessionFactory: manages the sessions,transactions of en

Session:

First level cache

Second Level Cache

Get vs load

CriteriaQuery

Save vs saveorupdate

Persist

Transient

Persistent

Detached

Sravanti Java

Class

Object

Eclipse

JVM

JRE

JDK

Data Types

Access Modifier

Packages

Abstraction

Encapsulation

Bindu

OOPS

Collections

Oevrloading Overriding

Data Types:

Primitive/Non Primitive

8 primitive

Int long double short byte Boolean char float

Wrapper class :

String

Access Modifiers:

Public :

Private:

Protected: With in the package as well as to the child classes outside the package.

Com.students

Student

Com.utility

SpecialNeedsStudents extends Students

Default:

OOPS:

Class:

Objects:

JVM

JRE

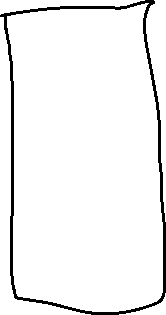
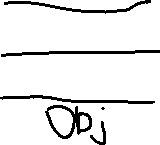
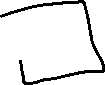
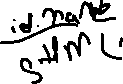
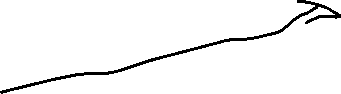
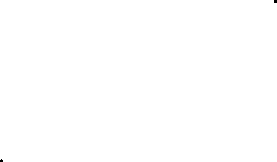
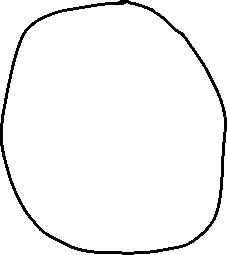
JDK

.java

.class – bytecode

Number is divisible by 3 print fizz ,by 5 print buzz and if its divible by both print fizzbuzz

Stack and Heap



package fizzbuzz;  
  
import java.util.Scanner;  
  
public class Fizzbuzz {  
  
     
    public static void main(String[] args) {  
        
        int number;  
       Scanner sc=new Scanner(System.in);  
        System.out.print ("Enter number: ");  
         number = sc.nextInt();  
          
         if(number %3 ==0 && number % 5==0){  
            System.out.println("Fizzbuzz");  
        }  
        else if(number%3==0){  
            System.out.println("Fizz");  
        }  
      
        else if(number %5 ==0){  
     System.out.println("Buzz");  
    }  
       
  
}  
}

Scanner sc = new Scanner(System.in);  
        System.out.println("Enter a number");  
        sc.nextInt();  
        int num = 0;  
        if (num % 3 == 0 && num % 5 == 0) {  
            System.out.println("FizzBuzz");  
        } else if (num % 3 == 0) {  
            System.out.println("Fizz");  
        } else if (num % 5 ==0) {  
            System.out.println("Buzz");  
        } else {  
            System.out.println(num);  
        }

Inheritance:

Java doesn’t support multiple inheritance directly.

Encapsulation: data and the code that acts on the data , we wrap it together as a single unit.

POJO: plain old java objects

Abstraction: hiding implementation ,

Abstract class

Interfaces

Polymorphism:

One name can be used in different forms.

Compile Time Polymorphism/Static/Method Overloading:

More than one method with same name but either the number of parameters or type of parameter must have to change.

Run Time/Dynamic/<Method Overriding: Inheritance.

Same signtaure

Int -4

Double-8

Constructor:

This

super

This : refer instance variable of current object.

Static

Final :

Abstraction:

Abstract class:

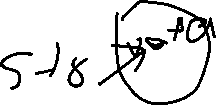
0-100%

Interfaces:

interface

String:

String pool:



Array

Collections

String reverse

Hello

olleh

Class Parent1

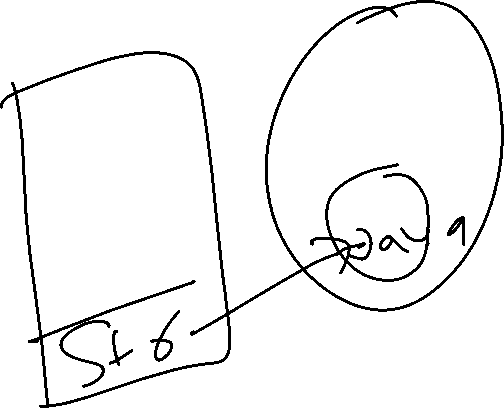
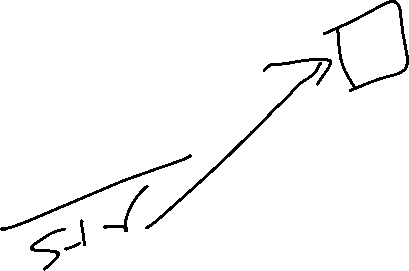
Void printaname

Class parent2

Vpid printname

Class child extends parent1 parent2

New child().printname()



Immutable Class :

StringBuilder- not synchronized, not safe

StringBuffer- synchronized, thread safe