**Java 2nd internal**

* 1. What is constructor? What are rules to create constructor.

Java supports a special type of method called a constructor that enables an object to initialize itself when it is created.

**Rules for creating Java constructor**

There are basically three rules defined for the constructor.

* + Constructor name must be same as its class name
  + Constructor must have no explicit return type. i.e Constructors do not have a return type — not even void
  + Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.
  1. What is Inheritance? What are different types of inheritance?

**Inheritance**

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

**Different Types of Inheritance**

* Single inheritance.
* Multi-level inheritance.
* Multiple inheritance.
* Multipath inheritance.
* Hierarchical Inheritance.
* Hybrid Inheritance.
  1. What is method overriding?

When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

1. Explain final variable and class.

* A variable declared with the final keyword is known as a final variable.
* It may be member variable or local variable.
* If you make any variable as final, you cannot change the value of a final variable(It will be constant).
* **Example:** public final double PI=3.142;

1. What is wrapper class.

A Wrapper class is **a class whose object wraps or contains primitive data types**. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types

The wrapper classes have a number of unique methods for handling primitive data types and objects.

1. What is finalizer method

Java supports a concept called finalization which is just opposite to initialization.

Java run-time is automatic garbage collecting system. It automatically frees up the memory resources used by the objects.

java.lang.Object.finalize()

Answer any four questions each carry five marks

* 1. Explain class, objects and methods with example.

Class

A class is a group of objects which have common properties.

It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

For example, we can consider a car as a class that has characteristics like steering wheels, seats, brakes, etc.

* An object is an instance of a class. Using class object is created.
* All these objects have a state and a behavior.
  + **State:** represents the data (value) of an object.
  + **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.

If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.

A Java method is **a collection of statements that are grouped together to perform an operation**. When you call the System. out. println() method, for example, the system actually executes several statements in order to display a message on the console.

**class** Student

{

**int** id;

 String name;

public void display()

{

System.out.println(s1.id+" "+s1.name);

}

**public** **static** **void** main(String args[])

{

  Student s1=**new** Student();

  s1.id=101;

  s1.name=“Priya";

  s1.diaplay();

 }

}

Output 101 Priya

* 1. Explain method overloading with example.

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

class TestOverloading

{

void add(int a,int b)

{

int c=a+b;

System.out.println(“result with 2 parameter”+c);

}

void add(int a,int b,int c)

{

int d=a+b+c;

System.out.println(“result with 3 parameter”+c);

}

public static void main(String[] args)

{

TestOverloading t=new TestOverloading();

t.add(11,11);

t.add(11,11,11);

}

}

Output

result with 2 parameter 22

result with 3 parameter 33

.

* 1. Explain multilevel inheritance with example.

Inheritance in java **occurs when a class extends a class that extends another class**. This is called multilevel Inheritance in java. For example, class C extends class B, and class B extends class A.

class Animal

{

void eat()

{

System.out.println("eating...");

}

}

class Dog extends Animal

{

void bark()

{

System.out.println("barking...");

}

}

class BabyDog extends Dog

{

void weep()

{

System.out.println("weeping...");

}

public static void main(String args[])

{

BabyDog d=new BabyDog();

d.weep();

d.bark();

d.eat();

}

}

* 1. Explain visibility control in JAVA.
* **Visibility Control**
* If we don't want the objects of class directly alter the value of a variable or access method. We can achieve this in Java by applying visibility modifiers to instance variable and methods. The visibility modifiers are also known as access modifiers.
* Java has four access modifiers:  
  1. public
* 2. default (Friendly)  
  3. private  
  4. protected
* **Public**
* The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.
* **Default**
* If you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package.
* **Private**
* The private access modifier is accessible only within the class.
* **Protected**
* The protected access modifier is accessible within package and outside the package but through inheritance only.
* The protected access modifier can be applied on the data member, method and constructor

public class A

{

protected void msg()

{

System.out.println("Hello");

}

}

package mypack;

import pack.\*;

class B extends A

{

public static void main(String args[])

{

B obj = new B();

obj.msg();

}

}

Output

Hello

1. What are different methods of String class? Give example.

String are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings.

public class StringMethodsDemo {

public static void main(String[] args) {

String targetString = "Java is fun to learn";

String s1= "JAVA";

String s2= "Java";

String s3 = " Hello Java ";

System.out.println("Char at index 2(third position): " + targetString.charAt(2));

System.out.println("After Concat: "+ targetString.concat("-Enjoy-"));

System.out.println("Checking equals ignoring case: " +s2.equalsIgnoreCase(s1));

System.out.println("Checking equals with case: " +s2.equals(s1));

System.out.println("Checking Length: "+ targetString.length());

System.out.println("Replace function: "+ targetString.replace("fun", "easy"));

System.out.println("SubString of targetString: "+ targetString.substring(8));

System.out.println("SubString of targetString: "+ targetString.substring(8, 12));

System.out.println("Converting to lower case: "+ targetString.toLowerCase());

System.out.println("Converting to upper case: "+ targetString.toUpperCase());

System.out.println("Triming string: " + s3.trim());

System.out.println("searching s1 in targetString: " + targetString.contains(s1));

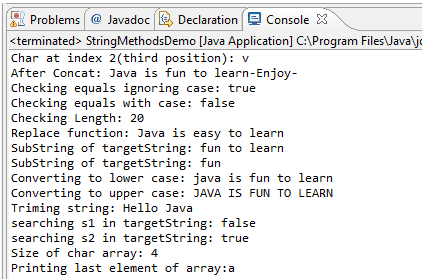
System.out.println("searching s2 in targetString: " + targetString.contains(s2));

char [] charArray = s2.toCharArray();

System.out.println("Size of char array: " + charArray.length);

System.out.println("Printing last element of array: " + charArray[3]);

}

} 

**Java 2nd internal**

* 1. What is constructor? What are rules to create constructor.

Java supports a special type of method called a constructor that enables an object to initialize itself when it is created.

**Rules for creating Java constructor**

There are basically three rules defined for the constructor.

* + Constructor name must be same as its class name
  + Constructor must have no explicit return type. i.e Constructors do not have a return type — not even void
  + Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.
  1. What is Inheritance? What are different types of inheritance?

**Inheritance**

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

**Different Types of Inheritance**

* Single inheritance.
* Multi-level inheritance.
* Multiple inheritance.
* Multipath inheritance.
* Hierarchical Inheritance.
* Hybrid Inheritance.
  1. What is method overriding?

When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

1. Explain final variable and class.

* A variable declared with the final keyword is known as a final variable.
* It may be member variable or local variable.
* If you make any variable as final, you cannot change the value of a final variable(It will be constant).
* **Example:** public final double PI=3.142;

1. What is wrapper class.

A Wrapper class is **a class whose object wraps or contains primitive data types**. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types

The wrapper classes have a number of unique methods for handling primitive data types and objects.

1. What is finalizer method

Java supports a concept called finalization which is just opposite to initialization.

Java run-time is automatic garbage collecting system. It automatically frees up the memory resources used by the objects.

java.lang.Object.finalize()

Answer any four questions each carry five marks

* 1. Explain class, objects and methods with example.

Class

A class is a group of objects which have common properties.

It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

For example, we can consider a car as a class that has characteristics like steering wheels, seats, brakes, etc.

* An object is an instance of a class. Using class object is created.
* All these objects have a state and a behavior.
  + **State:** represents the data (value) of an object.
  + **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.

If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.

A Java method is **a collection of statements that are grouped together to perform an operation**. When you call the System. out. println() method, for example, the system actually executes several statements in order to display a message on the console.

**class** Student

{

**int** id;

 String name;

public void display()

{

System.out.println(s1.id+" "+s1.name);

}

**public** **static** **void** main(String args[])

{

  Student s1=**new** Student();

  s1.id=101;

  s1.name=“Priya";

  s1.diaplay();

 }

}

Output 101 Priya

* 1. Explain method overloading with example.

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

class TestOverloading

{

void add(int a,int b)

{

int c=a+b;

System.out.println(“result with 2 parameter”+c);

}

void add(int a,int b,int c)

{

int d=a+b+c;

System.out.println(“result with 3 parameter”+c);

}

public static void main(String[] args)

{

TestOverloading t=new TestOverloading();

t.add(11,11);

t.add(11,11,11);

}

}

Output

result with 2 parameter 22

result with 3 parameter 33

.

* 1. Explain multilevel inheritance with example.

Inheritance in java **occurs when a class extends a class that extends another class**. This is called multilevel Inheritance in java. For example, class C extends class B, and class B extends class A.

class Animal

{

void eat()

{

System.out.println("eating...");

}

}

class Dog extends Animal

{

void bark()

{

System.out.println("barking...");

}

}

class BabyDog extends Dog

{

void weep()

{

System.out.println("weeping...");

}

public static void main(String args[])

{

BabyDog d=new BabyDog();

d.weep();

d.bark();

d.eat();

}

}

* 1. Explain visibility control in JAVA.
* **Visibility Control**
* If we don't want the objects of class directly alter the value of a variable or access method. We can achieve this in Java by applying visibility modifiers to instance variable and methods. The visibility modifiers are also known as access modifiers.
* Java has four access modifiers:  
  1. public
* 2. default (Friendly)  
  3. private  
  4. protected
* **Public**
* The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.
* **Default**
* If you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package.
* **Private**
* The private access modifier is accessible only within the class.
* **Protected**
* The protected access modifier is accessible within package and outside the package but through inheritance only.
* The protected access modifier can be applied on the data member, method and constructor

public class A

{

protected void msg()

{

System.out.println("Hello");

}

}

package mypack;

import pack.\*;

class B extends A

{

public static void main(String args[])

{

B obj = new B();

obj.msg();

}

}

Output

Hello

1. What are different methods of String class? Give example.

String are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings.

public class StringMethodsDemo {

public static void main(String[] args) {

String targetString = "Java is fun to learn";

String s1= "JAVA";

String s2= "Java";

String s3 = " Hello Java ";

System.out.println("Char at index 2(third position): " + targetString.charAt(2));

System.out.println("After Concat: "+ targetString.concat("-Enjoy-"));

System.out.println("Checking equals ignoring case: " +s2.equalsIgnoreCase(s1));

System.out.println("Checking equals with case: " +s2.equals(s1));

System.out.println("Checking Length: "+ targetString.length());

System.out.println("Replace function: "+ targetString.replace("fun", "easy"));

System.out.println("SubString of targetString: "+ targetString.substring(8));

System.out.println("SubString of targetString: "+ targetString.substring(8, 12));

System.out.println("Converting to lower case: "+ targetString.toLowerCase());

System.out.println("Converting to upper case: "+ targetString.toUpperCase());

System.out.println("Triming string: " + s3.trim());

System.out.println("searching s1 in targetString: " + targetString.contains(s1));

System.out.println("searching s2 in targetString: " + targetString.contains(s2));

char [] charArray = s2.toCharArray();

System.out.println("Size of char array: " + charArray.length);

System.out.println("Printing last element of array: " + charArray[3]);

}

} 