Java OOPs

Assignment Solutions

PW SKILLS

Q1. What are the differences between class and object ?

A1. Differences between class and object are following :

Constructors	Methods
1. Class is a blueprint from which objects are created. 2. Class is a group of similar type of objects. 3. Class is a logical entity. 4. Class is created using the class keyword. 5. Class is declared only once	1. Object is an instance of a class. 2. Object is a real world entity such as pencil, laptop etc. 3. Object is a physical entity. 4. Object is created using new keyword. 5. Object can be created many times.

Q2. Create a class named as student having attributes like String name, String rollNumber and String mobileNumber with a constructor accepting all the three attributes.

A2. Student class is created with name, rollNumber and mobileNumber as attributes.

JAVA Assignment48 SOL2

Q3. Create an object of the student class created above and print its attribute values as assigned by you.

A3. JAVA Assignment48 SOL3

Output:

```
Gaurav 2692 8978988767
```

Q4. Create a class named Animal having String name, String species and String genus along with an all argument constructor outside the main class.

A4. JAVA Assignment48 SOL4

Output:

```
Dog familiaris Canis
```

Q5. What are the benefits of making class outside the main class ?

A5. Benefits of making class outside the main class are following :

- We can reuse the code of the class as many times in main class by just making the object of the class.
- It increases code clarity and makes the code clean.
- It makes the code more object oriented and enhances the use of functions.
- It minimizes the code length and keeps the code minimal.

Q6. Why do we need a constructor ? Explain with the help of an example.

A6. We need a constructor because constructor is required to create an object of the class i.e an instance of the class. In the below code, if we had not defined a constructor or if java would not have initialized a default constructor then the code will not be able to create an instance of the class, resulting in a compile-time error.

JAVA Assignment49 SOL1

Q7. Make a Non-argument constructor of the class named Book which has attributes of string name and integer id and check output by printing the attribute values of the object after initializing it.

A7. JAVA Assignment49 SOL2

Output:

```
Constructor called
null
0
```

So, here we see that by default string has been initialized as null and integer equal to zero.

Q8. Why constructors in java cannot be static ?

A8. The constructors cannot be static in Java. When we declare a method as static, it means the method belongs to the class and not to a specific object. But We know that the constructor is always invoked to the reference of the objects. So, we can't make constructors static.

Q9. What is a destructor ? Do we have destructors in java ?

A9. Destructor is a special member function like constructor. Destructor destroys the class objects created by the constructor. Destructor is not present in java because it has its own garbage collector.

Q10. What is constructor chaining ? Explain with an example.

A10. Constructor chaining is defined as a technique of calling one constructor from another constructor. This technique is used with the help of 'this' keyword. As seen in the code, first constructor is called second and the second to third which means a chain is created hence the name. Let's assume in sequence constructors as 1st,2nd and 3rd. Then 1st is calling 2nd which is calling 3rd constructor and executes the code written in it.

JAVA Assignment49 SOL5

Output :

```
100
5
The Default constructor
```

Q11. Why is the main () method in Java always static?

A11. The main() method in Java is always defined as static because it must be called without creating an instance of the class in which it is defined. This is necessary because the JVM (Java Virtual Machine) calls the

main() method when the program starts, before any objects are created. If the main() method were not static, an instance of the class would need to be created before it could be called, which would defeat the purpose of the method.

Q12. Make a java class named Employee. Give it attributes like name, id and bossName. Make the bossName static. Write a function named print() which will print all three attributes.

A12. JAVA Assignment50 SOL2

Q13. As you have done in the previous question. Now, Make two objects of class Employee and assign them with unique attributes except bossName which will remain the same and then change the boss name of one object and then print e1.bossName and e2.bossName to see whether changing bossName of 1st employee has effect on bossName of second employee as well. Explain the output.

A13. JAVA Assignment50 SOL3

Output:

```
Rohit
Rohit
```

So, As we have changed the name of Varun's boss , but the name of Bhavit's boss has also changed along with it. That is the functionality of the static keyword. It is a class property and that's why it is not required to change it for every object to change it.

Q14. Make a Student class with attributes like String name and integer id. Use 'this' keyword to assign values assigned by user to object by making a parameterized constructor.

A14. JAVA Assignment50 SOL4

Output:

```
Prakhar 2765
Atul 2456
```

So, the constructor takes arguments and using this keyword assign it to different object's attributes.

Q15. What is the significance of 'this' keyword in java ?
A15. this keyword refers to the current object in a method or constructor.The most common use of this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).
this can also be used to:

- Invoke current class constructor
- Invoke current class method
- Return the current class object
- Pass an argument in the method call
- Pass an argument in the constructor call

Q16. Is it possible to overload a constructor?

Ans 16. Yes, it is possible to overload a constructor in Java.
Constructor overloading is a technique in which a class has multiple constructors with different parameter lists. Constructor overloading allows you to create multiple ways to construct an object of a class, each with a different set of parameters. This can be useful when you want to provide different options for creating an object, such as providing default values for some parameters or allowing the caller to specify different combinations of parameters.

Here is an example of constructor overloading in Java:

JAVA LP64 CODE4

In this example, the Person class has three constructors: one with all three parameters, one with only the name and age parameters, and one with only the name parameter. The second and third constructors use the first constructor to initialize the object, passing default values for the other parameters.

When creating an object of the Person class, you can use any of the three constructors depending on the information you have available.

Q17. What are Access modifiers in java? What types of access modifiers does java support?

A17. Access Modifiers are those modifiers that are used to restrict the visibility of classes, fields, methods, and constructors.
Java supports four types of access modifiers:
a) Private: Private members of a class can be accessed only within the class. It cannot be accessed from outside the class.
b) Default: Default members of a class are accessible within the same package due to visible only within the package. They cannot be accessed from outside the package.
c) Protected: Protected members of a class are visible within the package. Therefore, we can only access within the package but can be accessed to the subclasses outside the package through the inheritance only.
d) Public: Public members are visible anywhere. So, we can access it anywhere within or outside the package.

Q18. What is the difference between static and non-static classes?

A18. A nested class can be either static or non-static, in that case it's known as Inner class. The difference is that you can access a static class without creating an instance of the top level class but you would need an instance of the containing class to access a non-static nested class.

Q19. What is the final modifier, can you use it with classes?

A19. final keyword is not an access modifier but it's very important from an immutability and performance perspective. Yes, you can make a class final in Java, in that case you cannot extend it further. Final classes in Java cannot be subclassed.