Q: How to Create an Object in Java?

Answer: An object in Java is created using the `new` keyword.

Example:

java
Copy code
```
public class Example {
    int value;

    public Example(int value) {
        this.value = value;
    }

    public static void main(String[] args) {
        Example obj = new Example(10); // Creating an object of
Example class
    }
}
```

Q: What is the use of the `new` keyword in Java?

Answer: The `new` keyword in Java is used to create new objects. It allocates memory for the object and initializes it by calling the constructor.

Example:

java
Copy code
```
Example obj = new Example(10); // 'new' allocates memory and calls the
Example constructor
```

Q: What are the different types of variables in Java?

Answer: In Java, there are three main types of variables:

1. **Instance Variables**: Belong to instances of the class (objects).
2. **Static Variables**: Belong to the class and are shared among all instances.
3. **Local Variables**: Declared within a method and are accessible only within that method.

Example:

java
Copy code
```java
public class Example {
    int instanceVar; // Instance variable
    static int staticVar; // Static variable

    public void method() {
        int localVar = 10; // Local variable
    }
}
```

Q: What is the difference between Instance variables and Local variables?

Answer:

- **Instance Variables**:
    - Declared in a class but outside any method.
    - Each object of the class has its own copy.
    - Exists as long as the object exists.
- **Local Variables**:
    - Declared inside a method, constructor, or block.
    - Only accessible within that method, constructor, or block.
    - Exists only during the execution of the method, constructor, or block.

Example:

java
Copy code
```java
public class Example {
    int instanceVar; // Instance variable

    public void method() {
        int localVar = 10; // Local variable
    }
}
```

Q: In which area of memory is allocated for instance variables and local variables?

Answer:

- **Instance Variables**: Memory is allocated on the heap. Each object has its own copy of instance variables.
- **Local Variables**: Memory is allocated on the stack. They exist only during the method execution and are destroyed once the method completes.

Q: What is method overloading?

Answer: Method overloading in Java is a feature that allows a class to have more than one method with the same name, provided their parameter lists are different (different type, number, or both).

Example:

java
Copy code

```
public class Example {
    public void display() {
        System.out.println("No arguments");
    }

    public void display(int a) {
        System.out.println("One argument: " + a);
    }

    public void display(int a, int b) {
        System.out.println("Two arguments: " + a + ", " + b);
    }

    public static void main(String[] args) {
        Example obj = new Example();
        obj.display();
        obj.display(5);
        obj.display(5, 10);
    }
}
```