Java Concepts with Examples

1. Method: A block of code that performs a specific task.
   Example:
   ```
   public void greet() {
       System.out.println("Hello, World!");
   }
   ```

2. Overloaded Method: Multiple methods with the same name but different parameters.
   Example:
   ```
   public void display(int a) {
       System.out.println("Integer: " + a);
   }

   public void display(String s) {
       System.out.println("String: " + s);
   }
   ```

3. Class: A blueprint for creating objects.
   Example:
   ```
   class Car {
       String brand;
       int speed;
   }
   ```

4. Object: An instance of a class.
   Example:
   ```
   Car myCar = new Car();
   myCar.brand = "Tesla";
   myCar.speed = 120;
   ```

5. Constructor: Initializes objects.
   Example:
   ```
   class Car {
       String brand;

       Car() {
           brand = "Default Brand";
       }
   }
   ```

6. Overloaded Constructor: Multiple constructors with different parameters.

Example:

```
class Car {
    String brand;

    Car() {
        brand = "Tesla";
    }

    Car(String brandName) {
        brand = brandName;
    }
}
```

7. Inheritance: One class inherits from another.

Example:

```
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Dog barks");
    }
}
```

8. Polymorphism: Same method behaves differently in different classes.

Example:

```
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog barks");
    }
}
```

9. Public: Accessible from anywhere.

Example:

```
public class Demo {
```

```
    public int x = 10;
}
```

10. Private: Accessible only within the class.

```
Example:
class Demo {
    private int x = 10;
}
```

11. Protected: Accessible within package and subclasses.

```
Example:
class Parent {
    protected int data = 50;
}
```

12. Abstraction: Hiding implementation details.

```
Example:
abstract class Vehicle {
    abstract void start();
}

class Car extends Vehicle {
    void start() {
        System.out.println("Car starts with key");
    }
}
```

13. Getters and Setters: Access and modify private variables.

```
Example:
class Person {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String newName) {
        this.name = newName;
    }
}
```

14. ArrayList: A resizable array.

```
Example:
import java.util.ArrayList;
class Demo {
```

```
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Tejas");
        names.add("John");

        System.out.println(names.get(0));
    }
}
```

15. Wrapper Classes: Object representation of primitives.

```
Example:
class Demo {
    public static void main(String[] args) {
        Integer num = Integer.valueOf(10);
        Double pi = Double.valueOf(3.14);

        System.out.println(num);
        System.out.println(pi);
    }
}
```

Complete Code Example:

```
import java.util.ArrayList;

abstract class Animal {
    String name;

    Animal(String name) {
        this.name = name;
    }

    abstract void makeSound();
}

class Dog extends Animal {
    Dog(String name) {
        super(name);
    }

    @Override
    void makeSound() {
        System.out.println(name + " barks");
    }
```

```
}

class Person {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String newName) {
        this.name = newName;
    }
}

public class JavaConcepts {
    public static void main(String[] args) {
        Dog dog1 = new Dog("Buddy");
        dog1.makeSound();

        Person p = new Person();
        p.setName("Tejas");
        System.out.println("Person's name: " + p.getName());

        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(20);
        System.out.println("First number: " + numbers.get(0));

        Integer num = Integer.valueOf(100);
        System.out.println("Wrapped number: " + num);
    }
}
```