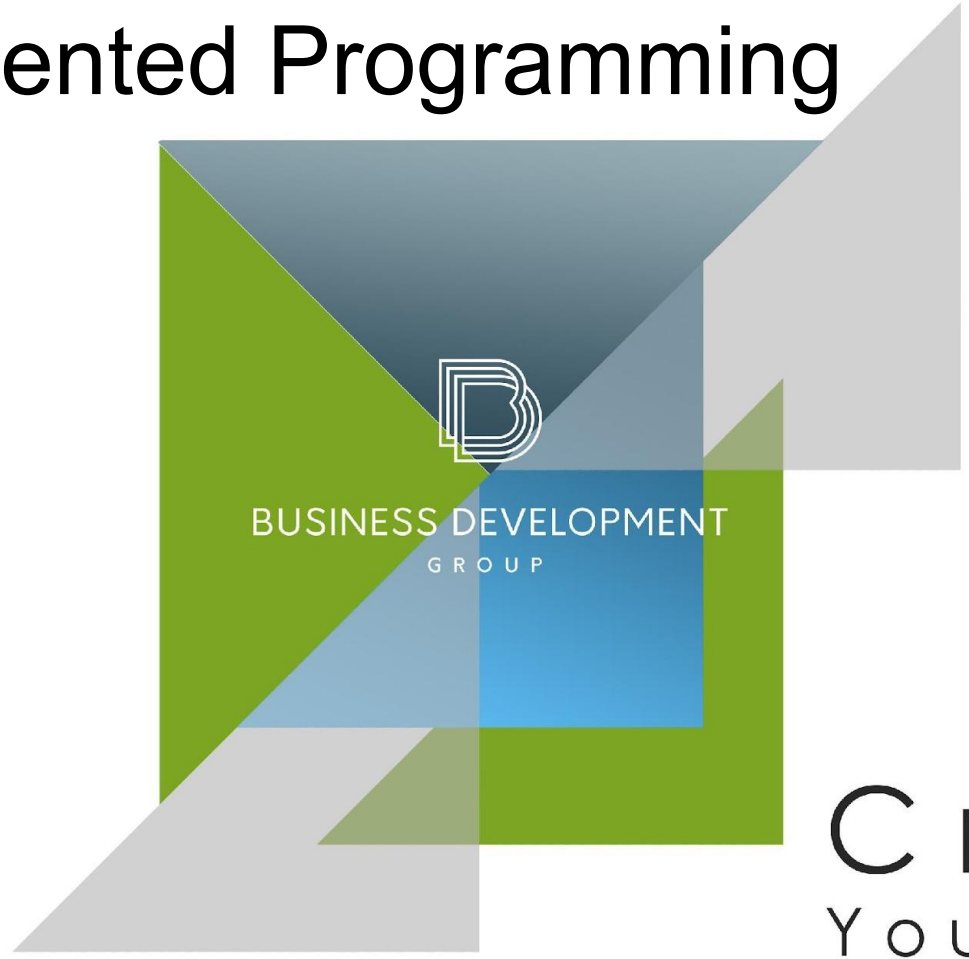
Object Oriented Programming



Create
Your Future

BUSINESS DEVELOPMENT
GROUP

Consider the following points

- Wrapper Classes
- Dates and Times
- Inheritance
- Encapsulation
- Polymorphism

BUSINESS DEVELOPMENT
GROUP

Wrapper Classes

Primitive type	Wrapper class	Example of constructing
boolean	Boolean	new Boolean(true)
byte	Byte	new Byte((byte) 1)
short	Short	new Short((short) 1)
int	Integer	new Integer(1)
long	Long	new Long(1)
float	Float	new Float(1.0)
double	Double	new Double(1.0)
char	Character	new Character(‘c’)

BUSINESS DEVELOPMENT
GROUP

Converting from a String

Wrapper class	Converting String to	Converting String to wrapper class
Boolean	Boolean.parseBoolean(“true”)	Boolean.valueOf(“true”)
Byte	Byte.parseByte(“1”)	Byte.valueOf(“2”)
Short	Short.parseShort(“1”)	Short.valueOf(“2”)
Integer	Integer.parseInt(“1”)	Integer.valueOf(“2”)
Long	Long.parseLong(“1”)	Long.valueOf(“2”)
Float	Float.parseFloat(“1”)	Float.valueOf(“2”)
Double	Double.parseDouble(“1”)	Double.valueOf(“2”)
Character	none	none

BUSINESS DEVELOPMENT
GROUP

Dates and Times

```
import java.time.*;

public class DatesAndTimes {

    public static void main(String[] args) {

        LocalDate.now();
        LocalTime.now();
        LocalDateTime.now();

        LocalDate date1 = LocalDate.of( year: 1997, month: 12, dayOfMonth: 25);
        LocalDate date2 = LocalDate.of( year: 1997,Month.DECEMBER, dayOfMonth: 25);
```

BUSINESS DEVELOPMENT
GROUP

Periods

```
LocalDate date = LocalDate.of( year: 1997,  month: 12,  dayOfMonth: 25);
LocalTime time = LocalTime.of( hour: 6,  minute: 15);
LocalDateTime dateTime = LocalDateTime.of(date, time);

Period period = Period.ofWeeks(3).ofMonths(1); // every month

date.plus(period);
dateTime.plus(period);
time.plus(period);
```

BUSINESS DEVELOPMENT
GROUP

Formatting

```
LocalDate date = LocalDate.of( year: 1997, Month.DECEMBER,  dayOfMonth: 25);
LocalTime time = LocalTime.of( hour: 11,  minute: 12,  second: 34);
LocalDateTime dateTime = LocalDateTime.of(date, time);

DateTimeFormatter shortFormat = DateTimeFormatter.
        ofLocalizedDateTime(FormatStyle.SHORT);

DateTimeFormatter mediumFormat = DateTimeFormatter.
        ofLocalizedDateTime(FormatStyle.MEDIUM);

dateTime.format(shortFormat);
mediumFormat.format(dateTime);
```

BUSINESS DEVELOPMENT
GROUP

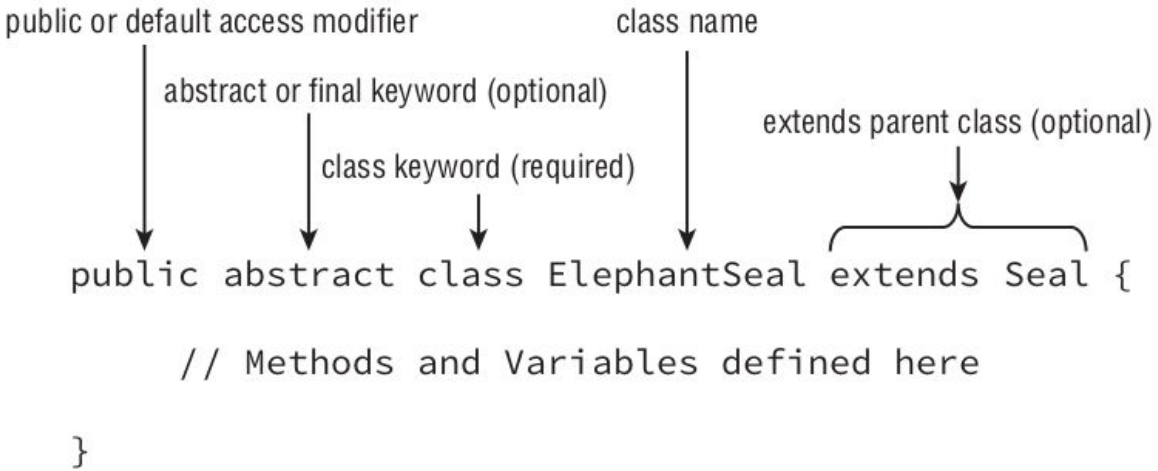
Parsing Dates and Times

```
DateTimeFormatter formatter = DateTimeFormatter.
        ofPattern("MM dd yyyy");

LocalDate date = LocalDate.parse( text: "12 25 1997", formatter);
LocalTime time = LocalTime.parse("11:22");
```

BUSINESS DEVELOPMENT
GROUP

Inheritance

Inheritance is the process by which the new child subclass automatically includes any public or protected primitives, objects or methods defined in the parent class.



BUSINESS DEVELOPMENT
GROUP

Encapsulation

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

```
public class Person {

    private int age;

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        if (age >= 0) {
            this.age = age;
        }
    }
}
```

BUSINESS DEVELOPMENT
GROUP

Access Modifiers

- public – from any class
- protected – from classes in the same package or subclasses
- default (package private) access - from classes in the same package
- private – from within the same class

BUSINESS DEVELOPMENT
GROUP

Polymorphism

Polymorphism in Java is a concept by which we can perform a single action in different ways.

There are two types of polymorphism in Java

- compile-time
- runtime

BUSINESS DEVELOPMENT
GROUP

Overloading

Should be different

- types of parameters
- numbers of parameters

Can be different

- access modifiers
- optional specifiers
- return types
- exception lists

```
public static short method(int i)
        throws Exception {
    return 1;
}

protected int overloadedMethod(long l)
        throws RuntimeException {
    return 2;
}
```

BUSINESS DEVELOPMENT
GROUP

Order Java uses to choose the right overloaded method

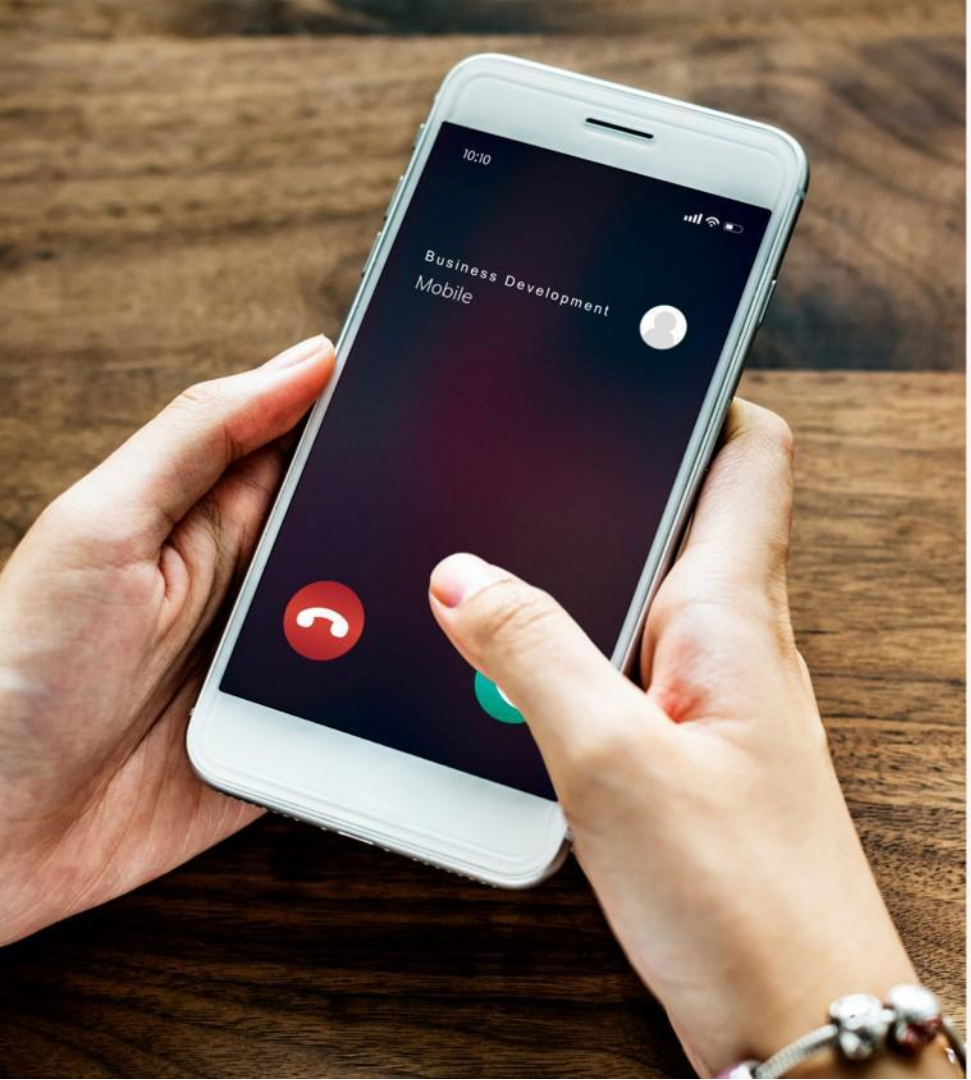
1) Exact match by type
2) Larger primitive type
3) Autoboxed type
4) Varargs

```
public static String glide(int i, int j) {
    return "First";
}

public static String glide(long i, long j) {
    return "Second";
}

public static String glide(Integer i, Integer j) {
    return "Third";
}

public static String glide(int... nums) {
    return "Fourth";
}

public static void main(String[] args) {
    glide( i: 1,  j: 2);
}
```

BUSINESS DEVELOPMENT
GROUP

Overriding methods

1) The method in the child class must have the same signature as the method in the parent class
2) The method in the child class must be at least as accessible or more accessible then the method in the parent class
3) If the method returns a value, it must be the same or a subclass of the method in the parent class, known as covariant return types
4) The method defined in the child class must be marked as static if it is marked as static in the parent class and otherwise.



Contact US

Phone: +374 55 201 209

E-mail: info@bdg.am

Address: Hr. Kochar 4

Web: www.bdg.am

https://www.facebook.com/bdg.trainings/

bdg.trainings

https://www.linkedin.com/in/bdg-trainings/