	POGRAMMING UNIT 4 - OBJECTS & CLASSES & INHERITANCES
0	11 Class Hierarchies muldianial 30
- 8	inhacits subclass inhacits from guardass subclass aveads superclass
(	All classes inherit from Object of the land of the lan
(	eq class Samoued extends Dog 3
	as class Fraction last motor of my
	All classes inherit from Object  e.g. class Samoyed extends Dog &  e.g. class Fraction extends Object  class Fraction extends Object  there a class inherit for only on the object of the objec
	When a class inherits from another it gets all of its fields method
-	When a class inherits from another it gets all of its fields of method.  When a method is redifined in a subclass, it is overniden.
C	DZ Terminology - printed sheets
0	13 Inheritance & Methods & Vaciables (both files)
	equals and tostring are provided by Object
	0 1/2/24 1
	> Student S = new Student (); Person (superclass)
	-> Person p = new Student(); Person (superclass)
	> Student S = new Student(); Person Superclass  → Person p = new Student(); Student Superclass  Object o = new Student(); Student Subclass
4	o Invalid
	4 Object 0 = new Object 0; L> Object 0 = new Student()
-	Students = 0
	Student s = (Student) o ; V
-	· Inheriting Fields:
	La Private fields are still private to a class.
	13 Private fields are still private to a class.  > To use/see/manipulate inherited private values, you must use
	> To use/see/manipulate inherited private values, you must use accessor and mutator methods
	> To use/see/manipulate inherited private values, you must use accessor and mutator methods
	13 Private fields are still private to a class.  > To use/see/manipulate inherited private values, you must use
0	13 Private fields are still private to a class.  To use/see/manipulate inherited private values, you must use accessor and mutator methods.  Protected makes the field visible to any class which inherits (subclean but not above in the hierarchy, nor autside the hierarchy.  4 Using Super
0	Is Private fields are still private to a class.  > To use/see/manipulate inherited private values, you must use accessor and mutator methods.  > Protected makes the field visible to any class which inherits (subclust not above in the hierarchy, nor autside the hierarchy.  4 Using Super  • It is possible to refer explicitly to the superclass of an object using significant superclass of super
0	13 Private fields are still private to a class.  To use/see/manipulate inherited private values, you must use accessor and mutator methods.  Trotected makes the field visible to any class which inherits (subclean but not above in the hierarchy, nor autside the hierarchy.  Using Super  It is possible to refer explicitly to the superclass of an object using since class Student extends Persons.
0	13 Private fields are still private to a class.  To use/see/manipulate inherited private values, you must use accessor and mutator methods.  Protected makes the field visible to any class which inherits (subclean but not above in the hierarchy, nor autside the hierarchy.  Using Super.  It is possible to refer explicitly to the superclass of an object using so class Student extends Persons.  Super():   happens in background.
0	13 Private fields are still private to a class.  To use/see/manipulate inherited private values, you must use accessor and mutator methods.  Protected makes the field visible to any class which inherits (subcloud not above in the hierarchy, nor autside the hierarchy.  Using Super  It is possible to refer explicitly to the superclass of an object using since class Student extends Persons.  Super ():   Asuper ():   happens in background 12 call super() at private String number;
0	Private fields are still private to a class.   To use/see/manipulate inherited private values, you must use accessor and mutator methods   Protected makes the field visible to any class which inherits (subcount not above in the hierarchy, nor outside the hierarchy.    Using Super   Using Super   Using Superclass of an object using size class Student extends Persons   Constructors:   Super () = happens in background   12 call super() at private String number;   public Student (String name String number) & Other Methods:
0	Private field are still private to a class.   To use/see/manipulate inherited private values, you must use accessor and mutator methods   Protected makes the field visible to any class which inherits (subcommutator methods but not above in the hierarchy, nor outside the hierarchy.    Using Super   It is possible to refer explicitly to the superclass of an object using so class Student extends Persons   Constructors:     Super () = happens in background   17 call super() at private String number;   public Student (String name String number) & Other Methods:     Super (name); // Like Person (name)   17 super, method Name)
0	Private fields are still private to a class.   To use/see/manipulate inherited private values, you must use accessor and mutator methods   Protected makes the field visible to any class which inherits (subcount not above in the hierarchy, nor outside the hierarchy.    Using Super   Using Super   Using Superclass of an object using size class Student extends Persons   Constructors:   Super () = happens in background   12 call super() at private String number;   public Student (String name String number) & Other Methods:
0	Private field are still private to a class.   To use/see/manipulate inherited private values, you must use accessor and mutator methods   Protected makes the field visible to any class which inherits (subcommutator methods but not above in the hierarchy, nor outside the hierarchy.    Using Super   It is possible to refer explicitly to the superclass of an object using so class Student extends Persons   Constructors:     Super () = happens in background   17 call super() at private String number;   public Student (String name String number) & Other Methods:     Super (name); // Like Person (name)   17 super, method Name)

05 Empl	nyee Assig	nment	:65	755	1	,	2	27.	F	30	) -	4	7	U	V	de	MA	A.	901	32		1
06 Poly	norphism												23	int	1.01	311		as	5	10		-
· Array		uperch	155	160	20 3	Str	re	an	un	sub	type	200			1	ts	- 21	179	lai	8	97	
. 7	-101		A	va	1	new	0	1001	0.	0	ME	B	vai	2	PIN	ew	BU	);	IA	0		
	7106		2	hois		60	1) &	-01	iesc	de	stx	E.	HAS	dorf	mD	, en	2,	m3	0,3	9		
6	- (m1) + overnie -m2 = inherita -m3 = new	ec.	1		-	m	6	- in	neni	red			00	13	1	3	طه		J.S.			
	-m3 & new		JI	1317	Vie	0		39	do	24	no	20	at	do	170	55	0/3					
bott examp	its helds	all of	35	B	Ji.	10	H	an	m	50	23	790	ni	55	AS		0	130	W	p		
0.0	stanivo s	131	255	130	SK	2	0	í.	60	A	red	3		en	SN	.5		118/1	W	0		
																				6.	-	_
									20	39	12	bo	JAL	70	100	100	ani	MI	Г	70		

## EXAMPLE 1 (Regular)

```
CLASS
public class Foof
      public void method1() {
System.out.println("foo1");
      public void method2 () {
              System. out. print In ("foo2");
       public String to String () {
              return "foo";
public class Bar extends Foo?
      public void method 2() }
               System.out. println("bar 2");
public class Baz extends Foo { public void method () {
              System. out. println ("baz 1");
        public String to String () {
return 11 baz";
public class Mumble extends Bazz
public void method2() }
System. out. println("mumble 2");
```

FOO

methodZ

```
method
method2
toString
            1302
            method
              toString
               Mumble
                method 2
```

```
CLIENT
Foo [] pity= { new Baz(), new Bar(), new Foo() };
for (int i=0; i<pity, length; i+1)}
    System.out.println (pity[i]);
     pity[i], method il);
pity[i], method 20;
     Systemout , printin (),
```

OUTPUT 11 baz object baz (Baz) baz / (Baz) foo 2 (FOOT) 1/ bar object foo (FOO 1) 100 ( FOOT) bar 2 (Bar) / mumble object baz (Baz 1) bazl (Baz1) mumble 2 (Mumble) // for object foo (Foo) fou 1 (F60) foo 2 (Foo)

trample 2 (Super/other calls) CLIENT CLASS Ham[] food: Enew Lamb), new HamD, new public class Lamb extends Ham? Spaml), new Yaml); public void bos furling i=0, ic food length, ++1){ System. out print ("Lamb b"); System. aut. print h (food [i]); Food[i]. a(); System. out. println(); food[i], b() public class Ham { public void a () } System. out. printino; Systemout. print ("Ham a"), System. out, print ("Ham b"); public void 60% 1/ Lamb Object Hom (Hant Ham a Lamb b (Hami, Lamb) public String tostring 08
return "Ham" Lamb b (Lamb) 1/ Ham Object public class Spam extends Yam? Ham a Hamb (Ham) public void b()? System. out. print ("Spam b"); Hamb (Ham) // Spam Object public class Yam extends Lamb? lam (4ant) public void all 2 System. cut, print ( Yam a 1); Yam a Ham a Spam b (4amt, Ham 7,5) super. al); Spam b (Spam) public String tostring() {
return yam", 11 Jan Object Jam (Yam) Yam a Ham a Lamb b (Yam, Hamt, Lends Lamb b (Lamb 1) Ham tostring Lamb am +6String Spam

## Introduction to Inheritance and Polymorphism

- Inheritance extends on an existing class
- Polymorphism is something you can do to classes that are being inherited
- You might want to extend a class by adding more specific items into the object
  - A car class might be extended to "BMW" class where it has BMW specific methods and variables
  - o Or a Circle class might be extended to "Disk" and accepts the input of height.

## Inheritance

- Inheritance: when you extend classes from a "super class" to make it do more specific
  things
  - o "allows a class to define a specialized type of an already existing class"
  - Is-a relationship: an extended class "is a" type of another class
    - BMW "is a" type of a car class. Disk "is a" type of circle class.
- Implementation:
  - o extends: a keyword you must use to declare a "sub class"
    - this keyword will make it refer to the super class
    - public class <name> extends <superclass name> { ... }
  - base class: or "super class: is the class which this sub class is based on
  - derived class: or "sub class" often contains overridden methods or new data from the base class
  - o super: is a keyword used to access methods in the base class
    - Ex. super(r) from the BMW class will call the constructor of the super class (car), and pass variable r into it.
  - Members that are private cannot be accessed by derived classes
    - Hence, accessor methods are used to get values (ie getWeight(), getRadius())
  - o Inherited Methods can be called directly.
    - If M3 is the object of BMW class, and you call carStart() from client code. If BMW class doesn't have it's own carStart() method, but Car class does, since BMW inherited the Car class, the client code doesn't care and still executes carStart().

## **Polymorphism**

- Polymorphism: an OOP property where objects have the ability to assume different types.
  - o Polymorphism is based on inheritance
  - Since a subclass is derived from a super class, a super class object can reference an object of the subclass

ie Car honda;

BMW 320i = new BMW(<arguments>); honda = 320i //honda now references 320i

- Since BMW is inherited from Car class, any object made from Car can be "morphed" into another object made by a subclass of Car, in this case, a BMW 320i object.
- Overridden methods from the super class can be called from the sub class.
- From the example, the method toString() originally in Car was overridden in BMW. Even though honda was made by Car class, then later morphed into a BMW object, I can still call the toString() method overridden by BMW.