EGCI 213 Midterm (open book + calculator, no electronic device)

** Student uniform + student card

** Big and clear handwriting

1. **Variable** ➔ lifetime & program's address space, scope

• Local variable	stack frame of method (in runtime stack)		local scope
• Static member	static area	allocated to class	visibility level
• Non-static member	heap	allocated to object	visibility level
• Final variable	local, static, or non-static that is constant		

2. **Array of objects** ➔ creation, usage

3. **Class & object**

• Member variable	static, non-static
• Member method	static, non-static
• Visibility level	public > protected > default (no prefix) > private
• Constructor	default constructor, constructor chain
• Overloading	methods/constructors in the same class ➔ different parameters

4. **Inheritance**

• From parent to child	all parent's components except constructor
• java.lang.Object	default inheritance
• Overriding	methods in parent & child ➔ same parameters, >= visibility in child
• Using parent's member	static ➔ class.member
	non-static ➔ super.member (parent's method must be concrete)
• Final class	no inheritance
• Final method	no overriding

```
class Man extends java.lang.Object {
    public Man ( )                  { super( ); }
    public Man (String name)        { this( ); other instructions }
}
```

5. **Abstraction** ➔ pointer type allowed, object creation not allowed

• Abstract class	constructor, abstract (non-static) method, concrete (static & non-static) method static & non-static variable, any visibility	
• Interface	abstract (non-static) method, static final variable, public visibility	
• Child class	class Child extends AbstractParent	single parent class
	class Child implements MomInterface, DadInterface	multiple parent interfaces
• Inheritance	class extends class, interface extends interface, class implements interface	

6. **Polymorphism**

• Pointer rules	Parent p = new Child ( );	Object p = new Child ( );
	((Child) p).childMethod( );	
	if (p instanceof Child)	System.out.println("Child object");
• Polymorphism rules	all classes have polymorphic method, use parent pointer for all objects	
By class	all classes extend same parent class	parent pointer = parent class
By interface	all classes implement same interface	parent pointer = interface

7. **Generic & sorting**

- ArrayList<E>   E = Child   keep 1 type of objects
  E = Parent, ParentInterface   keep >1 types of objects for polymorphism

  Usage   add ( ), get (int index), size ( )

`Man [ ] allPersons = new Man[3];`	`ArrayList<Man> allPersons = new ArrayList<Man>();`
	`ArrayList<Man> allPersons = new ArrayList<>();`

`allPersons[0] = new Man(…)`
`allPersons[1] = new Woman(…);`
`allPersons[2] = new Man(…);`
	`allPersons.add( new Man(…) );`
	`allPersons.add( new Woman(…) );`
	`allPersons.add( new Man(…) );`

`Arrays.sort( allPersons );`
`for(Man m : allPersons)   m.speak();`
	`Collections.sort( allPersons );`
	`for(Man m : allPersons)   m.speak();`

- Sorting requirement   class implements Comparable ➔ method compareTo to return -1, 0, 1

```
class Man implements Comparable<Man> {
    public int compareTo (Man other) {
        if (this.age < other.age)          return -1;     // this is placed before other
        else if (this.age > other.age)     return 1;      // this is placed after other
        else                               return 0;
    }
}
```

8. **Exception**

- Checked exception   extend Exception ➔ IOException, FileNotFoundException, InterruptedException
- Unchecked exception   extend RuntimeException ➔ NullPointerException, NumberFormatException, ArrayIndexOutOfBoundsException
- Actual type of exception

```
try { … }
catch(Exception e)  { System.err.println( e.getClass().getName() ); }
```

- Method throwing exception   checked vs. unchecked exception
- Propagation   checked vs. unchecked exception
- Try-catch-finally

9. **Basic coding** ➔ Maven project (src/main/java/…)

- Folder structure & package instruction
- File name & class name
- File path

Questions

Q1. Given a program

- Add proper package instruction, class/file name, file path
- See 9

Q2. Given a program

- Trace the program ➔ output = ? ** only the first n lines will be graded
- Explain properties of some variables or methods in the program
- See 1, 2, 3

Q3. Given interfaces, abstract classes, concrete classes

- Explain inheritance rules ➔ what if some instructions/keywords are added, removed, changed ?
- Explain constructor chain ➔ what if some instructions/keywords are added, removed, changed ?
- Write a few instructions in main method ➔ using ArrayList, object creation, polymorphism, sorting
- See 4, 5, 6, 7

Q4. Given a program with possible exceptions

- Trace the program ➔ why an exception occurs (by which instruction)
  where it is handled (by which catch)
- Output from the program ➔ normal results (System.out) + exception messages (System.err)
  ** only the first n lines will be graded
- What if some instructions/exceptions are added, removed, changed ?
- See 8

** In all questions, when explain the existing / lacking of certain property that causes error (or no error):

- Don't just give a keyword e.g. because this variable is final
- But explain what this keyword means e.g. this variable is final, which means its value can't be changed