USN | | | | | | | | | | |

Internal Assessment Test 1 – December 2021

Sub:	Software Engineering					Sub Code:	18CS35	Branch:	ISE	
Date:	16/12/2021	Duration:	90 min's	Max Marks:	50	Sem/Sec:	III / A,B and C		OBE	
Answer any FIVE FULL Questions								MARKS	CO	RBT
1	What is software engineering? Describe software engineering code of ethics?							10	CO2	L1
2	With a neat diagram, explain the waterfall model of software development process?							10	CO1	L2
3	Consider the use case insulin pump control system? Construct the model and explain it.							10	CO1	L3
4	Explain the following important terms with example? i) Identity ii) Classification iii) Inheritance iv) Polymorphism							10	CO1	L2
5	What is Object Oriented Development? Explain different stages of object oriented development							10	CO3	L2
6	What are Classes and objects? Write and explain UML notation for objects and classes, values and attributes.							10	CO3	L2

Faculty Signature CCI Signature HOD Signature

USN | | | | | | | | | | |

Internal Assessment Test 1 – December 2021

Sub:	Software Engineering					Sub Code:	18CS35	Branch:	ISE	
Date:	16/12/2021	Duration:	90 min's	Max Marks:	50	Sem/Sec:	III / A,B and C		OBE	
Answer any FIVE FULL Questions								MARKS	CO	RBT
1	What is software engineering? Describe software engineering code of ethics?							10	CO2	L1
2	With a neat diagram, explain the waterfall model of software development process?							10	CO1	L2
3	Consider the use case insulin pump control system? Construct the model and explain it.							10	CO1	L3
4	Explain the following important terms with example? i) Identity ii) Classification iii) Inheritance iv) Polymorphism							10	CO1	L2
5	What is Object Oriented Development? Explain different stages of object oriented development							10	CO3	L2
6	What are Classes and objects? Write and explain UML notation for objects and classes, values and attributes.							10	CO3	L2

Faculty Signature CCI Signature HOD Signature

1c

Scheme of Evaluation
Internal Assessment Test 1 – May 2021

Sub:	Software Engineering							**Code:**	18CS35
Date**:**	16/12/2021	Duration:	90mins	Max Marks:	50	**Sem:**	III	**Branch:**	ISE

Note: **Answer Any five full questions.**

Question #		**Description**	**Marks Distribution**		**Max Marks**
1		Software engineering is an engineering discipline that is concerned with all aspects of software production Explanation of software engineering code of ethics Honesty and Integrity. Confidentiality Competence Intellectual property rights Computer misuse	2M 8M	10M	10M
2		The waterfall model takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on. **Diagram** **Explanation**	2M 4M 4M	10M	10M
3		An insulin pump is a medical system that simulates the operation of the pancreas (an internal organ). **Diagram** **Steps** **Example**	2M 4M 4M	10M	10M

4		Explain about characteristics of Object-Oriented Development with example **Explanation** **Example**	 10M	10M	10M
5		Definition of Object-Oriented development **Explanation** **Diagram**	2M 4M 4M	10M	10M
6		Definition of Classes and Objects UML Notation for Classes and Objects UML notation for Values and Attributes	2M 4M 4M	10M	10M

Scheme of Evaluation Internal Assessment Test 1 – DEC 2021

Sub:	Software Engineering							Code:	18CS35
Date:	16/12/2021	Duration:	90mins	Max Marks:	50	Sem:	III	Branch:	ISE

Note: **Answer Any full five questions**

Q. 1 What is software engineering? Explain software engineering code of ethics?

Software engineering is an engineering discipline that is concerned with all aspects of software production

It goes without saying that you should uphold normal standards of honesty and integrity.

1. *Confidentiality* You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
2. *Competence* You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.
3. *Intellectual property rights* You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.
4. *Computer misuse* You should not use your technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses or other malware).

Software engineers shall adhere to the following Eight Principles:

1. PUBLIC — Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER — Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT — Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT — Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT — Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION — Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES — Software engineers shall be fair to and supportive of their colleagues.
8. SELF — Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Q.2 With a neat diagram, explain the waterfall model of software development process?

The waterfall model takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.

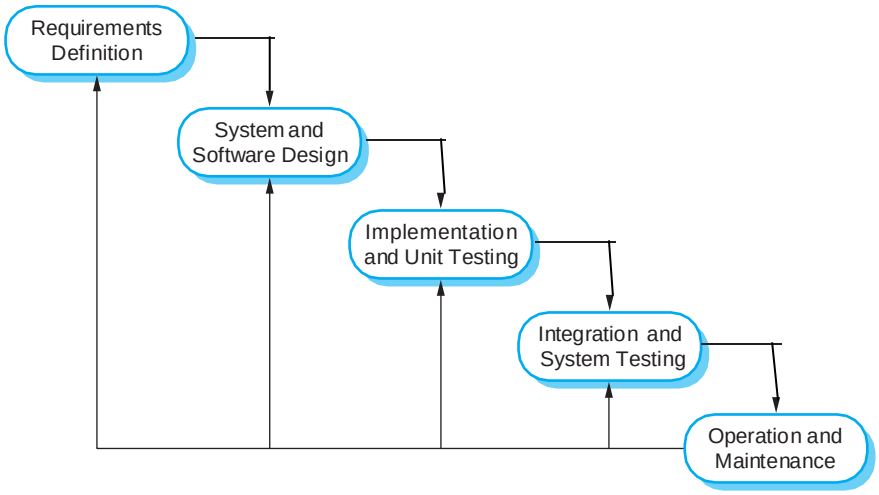


Figure: The waterfall model

The principal stages of the waterfall model directly reflect the fundamental development activities:

Requirement's analysis and definition the system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

System and software design the systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

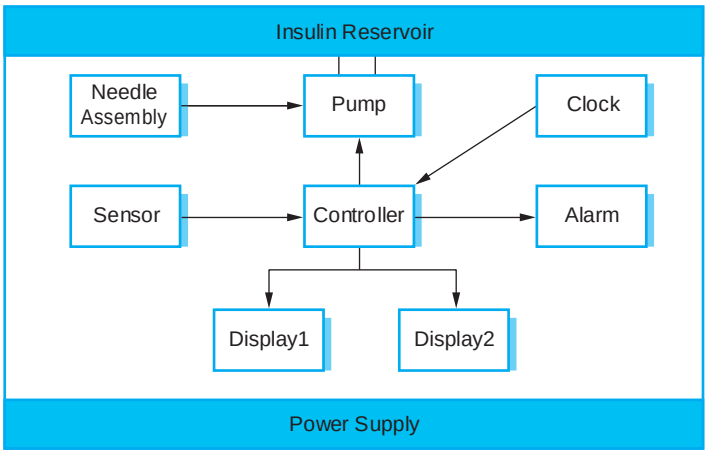
Implementation and unit testing During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

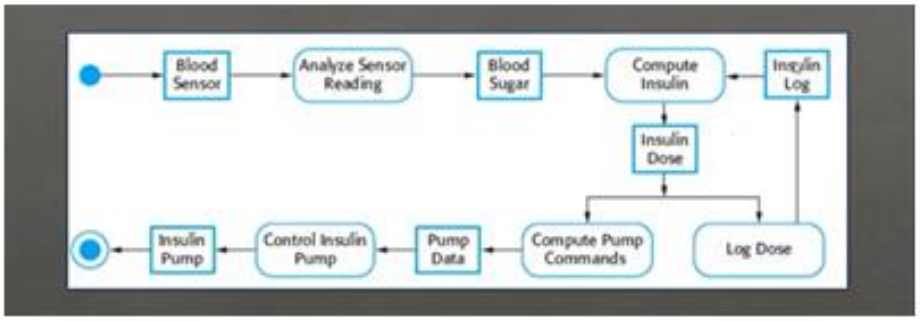
Integration and system testing the individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

Operation and maintenance Normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

In principle, the waterfall model should only be used when the requirements are well understood and unlikely to change radically during system development. However, the waterfall model reflects the type of process used in other engineering projects. As is easier to use a common management model for the whole project, software processes based on the waterfall model are still commonly used.

Q.3 With a neat diagram, explain the insulin pump control system.





Current advances in developing miniaturized sensors have meant that it is now possible to develop automated insulin delivery systems. These systems monitor blood sugar levels and deliver an appropriate dose of insulin when required. Insulin delivery systems like this already exist for the treatment of hospital patients. In the future, it may be possible for many diabetics to have such systems permanently attached to their bodies.

A software-controlled insulin delivery system might work by using a micro- sensor embedded in the patient to measure some blood parameter that is proportional to the sugar level. This is then sent to the pump controller. This controller computes the sugar level and the amount of insulin that is needed. It then sends signals to a miniaturized pump to deliver the insulin via a permanently attached needle.

Essential high-level requirements:

- The system shall be available to deliver insulin when required.
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

4. Explain the following important terms with example?
i) Identity ii) Classification iii) Inheritance iv) Polymorphism

Identity means that data is organized into discrete, distinguishable entities called objects.
An object has:

- *state* - descriptive characteristics
- *behaviors* - what it can do (or what can be done to it)

- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

Software objects model read-world objects or abstract concepts

- dog, bicycle, Bank account

Real-world objects have states and behaviors

- Dogs' states: name, color, breed, hungry
- Dogs' behaviors: barking fetching

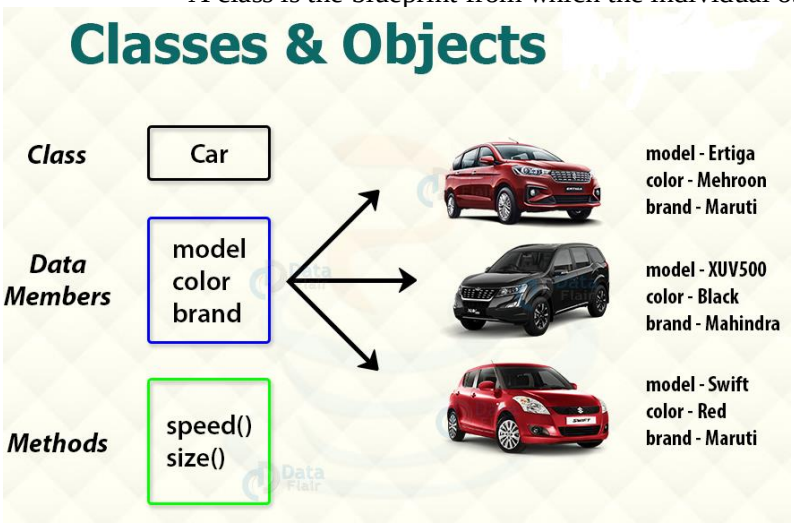
Objects have three responsibilities:

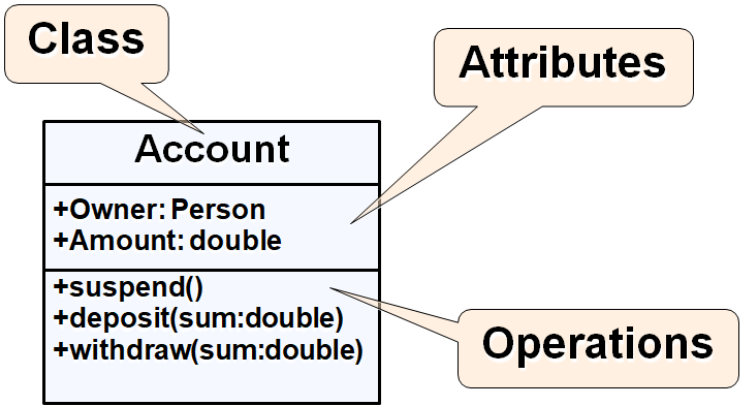
What they know about themselves – (e.g., Attributes)
What they do – (e.g., Operations)
What they know about other objects – (e.g., Relationships)

Classification

It means that objects with same data structure (attribute) and behavior (operations) are **grouped into a class.**

- A *class* is simply a representation of a type of *object*. It is the blueprint/ plan/ template that describe the details of an *object.*
- A class is the blueprint from which the individual objects are created.

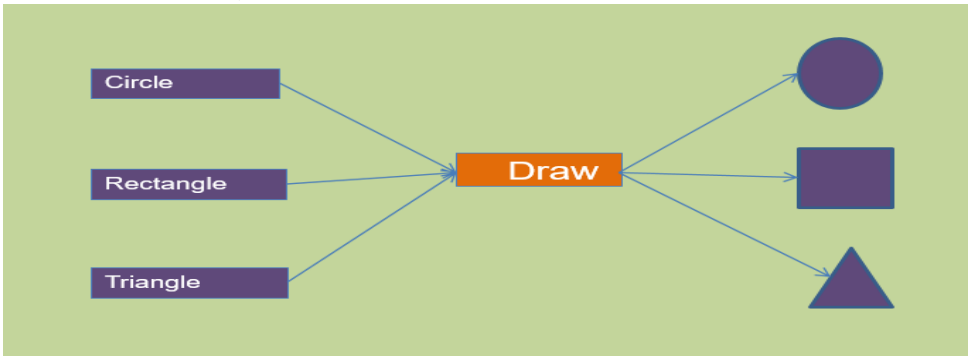




Polymorphism

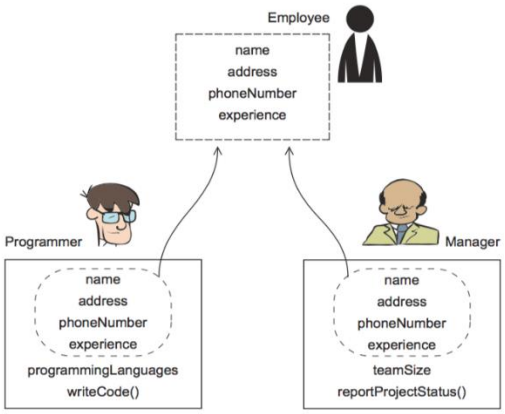
- It means that the same operation (i.e. action or transformation that the object performs) may behave differently on different classes.
- Ability of different objects to response same message in different ways.
- Ability of an object to take on multiple forms.

- In a programming language, class objects belonging to the same hierarchical tree may have functions with the same name, but with different behaviors.

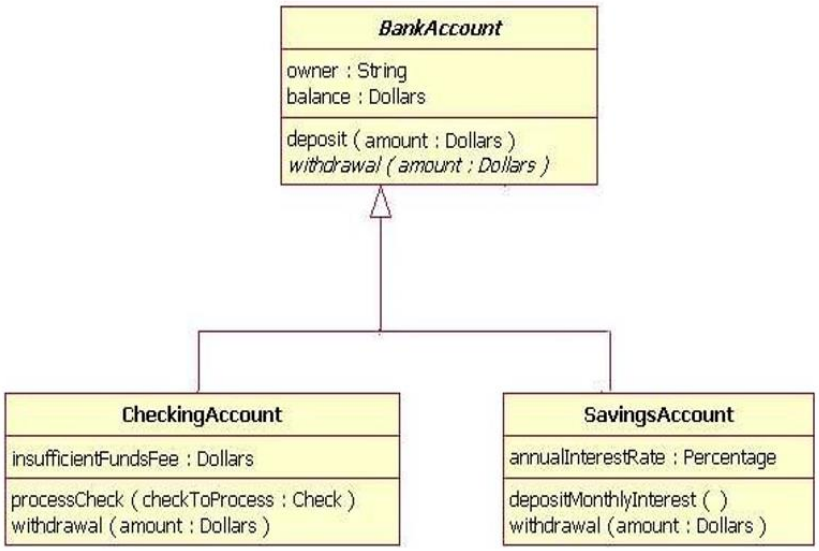


iv)Inheritance:

- It is the sharing of attributes and operations among classes based on a hierarchical relationship.
- Subclasses can be formed from broadly defined class.
- Each subclass incorporates or inherits all the properties of its super class and adds its own unique properties.



Inheritance



5. What is Object Oriented Development? Explain different stages of object oriented Development

Object Oriented Development

The essence of OO Development is the identification and organization of application concepts, rather than their final representation in a programming language.

It's a conceptual process independent of programming languages. OO development is fundamentally a way of thinking and not a programming technique

Object-Oriented Methodology

- The process for OO development and graphical notation for representing OO concepts consists of building a model of an application and then adding details to it during design.

The methodology has the following stages:

- 1.**System conception** : Software development begins with business analysts or users conceiving an application and formulating tentative requirements.
- **2.Analysis** : The analyst must work with the requestor to understand the problem, because problem statements are rarely complete or correct.

The analysis model is a precise abstraction of what the desired system must do, not how it will be done.

- It should not contain implementation decisions
- Ex: identifying the classes, its attributes and operations.

The analysis model has 2 parts:

- **2.a Domain model -** a description of the real-world objects reflected within the system.

Example for Domain objects for a stock broker.

stock, bond, trade and commission.

- **2.b Application model -** a description of the parts of the application system itself that are visible to the user.

Ex: Application objects might control the execution of trades and present the results.

Application experts who are not programmers can understand and criticize a good model.

- **3.System design:** The development teams devise a high – level strategy – the system architecture for solving the application problem.
    - Establish policies that will serve as a default for the subsequent portion of design
    - Decide performance characteristics to optimize the problems.
- **4.Class design:** The class designer adds details to the analysis model in accordance with the system design strategy.
- The focus of class design is the data structures and algorithms needed to implement each class.
- **5. Implementation:** Implementers translate the classes and relationships developed during class design into particular programming language, database or hardware.

During implementation, it is important to follow good software engineering practice so that traceability to the design is apparent and so that the system remains flexible and extensible.

Three kinds of models to describe a system from different viewpoints.,

- 1**. Class Model**—for the objects in the system & their relationships. It describes the static structure of the objects in the system and their relationships. Class model contains class diagrams- a graph whose nodes are classes and arcs are relationships among the classes.
- 2. **State model**—for the life history of objects. It describes the aspects of an object that change over time. It specifies and implements control with state diagrams-a graph whose nodes are states and whose arcs are transition between states caused by events.
- 3**. Interaction Model**—for the interaction among objects. It describes how the objects in the system co-operate to achieve broader results. This model starts with use cases that are then elaborated with sequence and activity diagrams.
    - Use case – focuses on functionality of a system – i.e what a system does for users.
    - Sequence diagrams – shows the object that interact and the time sequence of their interactions.

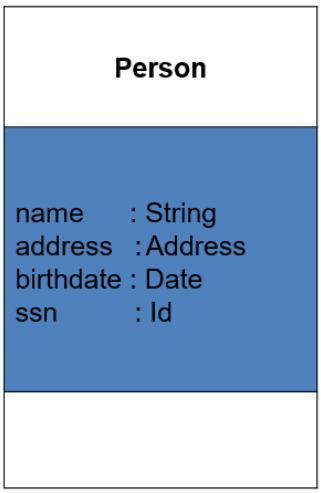
- Activity diagrams – elaborates important processing step

5. What are classes and objects? Write and explain UML notation for objects and classes, values and attributes.

Class diagrams provide a graphic notation for modeling classes and their relationships, thereby describing possible objects.

- A Class Diagram is a diagram describing the structure of a system
- Shows the system's
  - Classes
  - Attributes
  - Operations (or methods),
  - Relationships among the classes

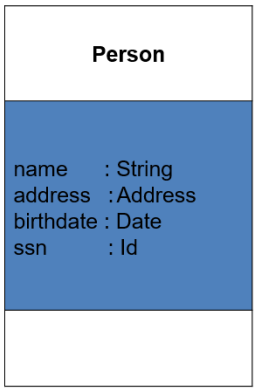
Class Attributes



An *attribute* is a named property of a class that describes the object being modeled.

In the class diagram, attributes appear in the second compartment just below the name-compartment.

Class Attributes



An *attribute* is a named property of a class that describes the object being modeled.

In the class diagram, attributes appear in the second compartment just below the name-compartment.

Attributes are usually listed in the form:

attributeName : Type

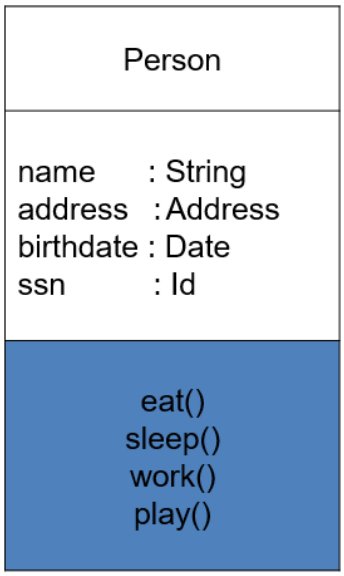
A *derived* attribute is one that can be computed from other attributes, but doesn't actually exist.

For example,a Person's age can be computed from his birth date. A derived attribute is designated by a preceding '/' as in:

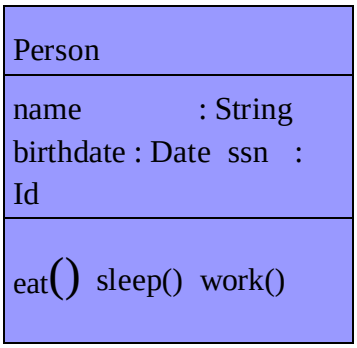
/ age : Date

Attributes can be:
+ public
 # protected
- Private
/ derived

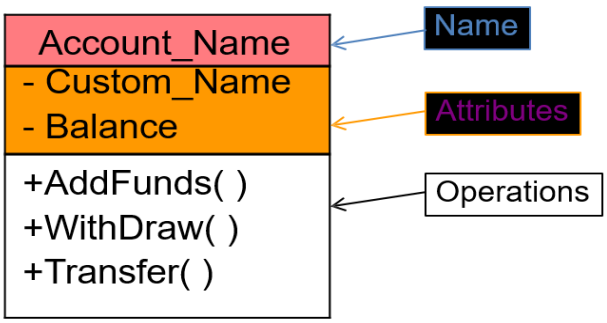
Class Operations

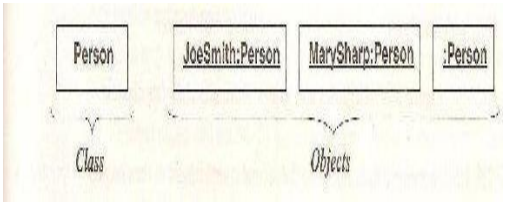


Operations describe the class behavior and appear in the third compartment.



An example of Class



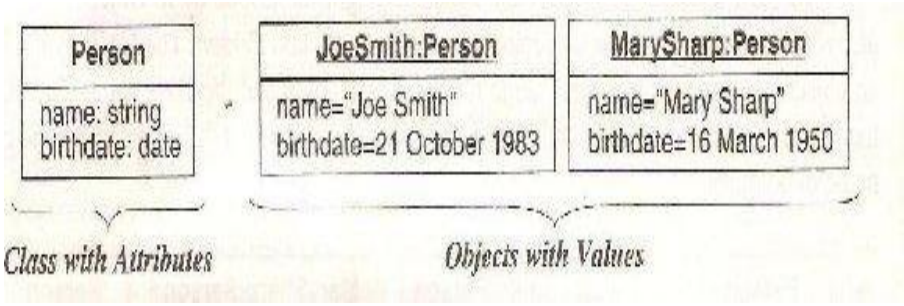


- **Conventions used (UML):**
- UML symbol for both classes and objects is box.
- Objects are modeled using box with object name followed by colon followed by class name, Both the names are underlined.
- Use boldface to list class name, center the name in the box and capitalize the first letter. Use singular nouns for names of classes.
- To run together multiword names (such as JoeSmith), separate the words With intervening capital letter.

Values and Attributes:

- Value is a piece of data
- Attribute is a named property of a class that describes a value held by each object of the class.

E.g. Attributes: Name, bdate, weight. Values: JoeSmith, 21 October 1983, 64.



List attributes in the 2nd compartment of the class box.

A colon precedes the type, an equal sign precedes default value.

Show attribute name in regular face, left align the name in the box and use small case for the first letter.

Similarly we may also include attribute values in the 2nd compartment of object boxes with same conventions.