Abstract Class

- A class that is declared using “abstract” keyword is known as abstract class.
- It can have abstract methods(methods without body) as well as concrete methods (regular methods with body).
- A normal class(non-abstract class) cannot have abstract methods.
- But, if a class has at least one abstract method, then the class must be declared abstract.
- If a class is declared abstract, it cannot be instantiated.

Abstract Class

- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Abstract Method

- Inheritance allows a sub-class to override the methods of its super-class.
- In fact, a super-class may altogether leave the implementation details of a method and declare such a method abstract:
- abstract type name(parameter-list);
- Two kinds of methods:
- 1) concrete – may be overridden by sub-classes
- 2) abstract – must be overridden by sub-classes
- It is illegal to define abstract constructors or static methods.

Abstract Class

- A class that contains an abstract method must be itself declared abstract:
- abstract class abstractClassName {

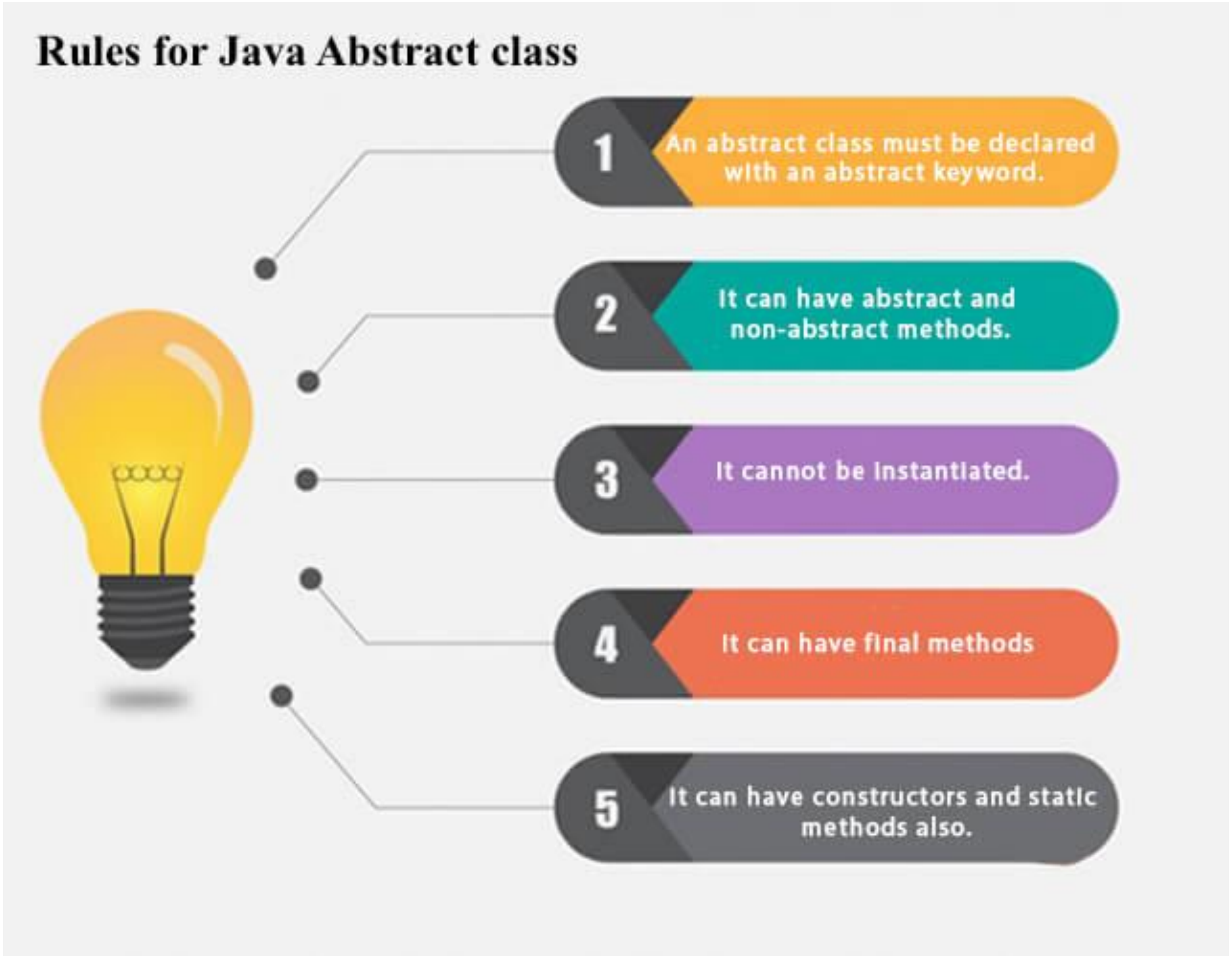
  abstract type methodName(parameter-list) {

  …

  }

  …
- }
- An abstract class has no instances - it is illegal to use the new operator:
- ~~abstractClassName a = new abstractClassName();~~
- It is legal to define variables of the abstract class type.

Abstract Class



Rules for Java Abstract class

1 An abstract class must be declared with an abstract keyword.

2 It can have abstract and non-abstract methods.

3 It cannot be instantiated.

4 It can have final methods

5 It can have constructors and static methods also.

Abstract Sub-Class

- A sub-class of an abstract class:

1) implements all abstract methods of its super-class, or

2) is also declared as an abstract class

-
  ```
  abstract class A {
      abstract void callMe();
  }
  ```

-
  ```
  abstract class B extends A {
      int checkMe;
  }
  ```

Abstract and Concrete Classes

- Abstract super-class, concrete sub-class:

```
•  abstract class A {
          abstract void callme();
          void callmetoo() {
                    System.out.println("This is a concrete method.");
          }
•  }
•  class B extends A {
          void callme() {
                    System.out.println("B's implementation.");
          }
•  }
```

Abstract and Concrete Classes

- Calling concrete and overridden abstract methods:

```
class AbstractDemo {
    public static void main(String args[]) {
        B b = new B();
        b.callme();
        b.callmetoo();
    }
}
```

Example: Abstract Class

- Figure is an abstract class; it contains an abstract area method:
- abstract class Figure {

```
    double dim1;
    double dim2;
    Figure(double a, double b) {
        dim1 = a; dim2 = b;
    }
    abstract double area();
```

- }

Example: Abstract Class

- Rectangle is concrete – it provides a concrete implementation for area:

```
class Rectangle extends Figure {
	Rectangle(double a, double b) {
		super(a, b);
	}
	double area() {
	System.out.println("Inside Area for Rectangle.");
		return dim1 * dim2;
	}
```

- }

Example: Abstract Class

- Triangle is concrete – it provides a concrete implementation for area:
- class Triangle extends Figure {

```
        Triangle(double a, double b) {
                super(a, b);
        }
        double area() {
                System.out.println("Inside Area for Triangle.");
                return dim1 * dim2 / 2;
        }
```

- }

Example: Abstract Class

- Invoked through the Figure variable and overridden in their respective subclasses, the area() method returns the area of the invoking object:
- class AbstractAreas {

```
      public static void main(String args[]) {
            Rectangle r = new Rectangle(9, 5);
            Triangle t = new Triangle(10, 8);
            Figure figref;
            figref = r; System.out.println(figref.area());
            figref = t; System.out.println(figref.area());
      }
```

- }

Abstract Class References

- It is illegal to create objects of the abstract class:
- Figure ~~f = new Figure(10, 10);~~
- It is legal to create a variable with the abstract class type:
- Figure figref;
- Later, figref may be used to assign references to any object of a concrete sub-class of Figure (e.g. Rectangle) and to invoke methods of this class:
- Rectangle r = new Rectangle(9, 5);
- figref = r; System.out.println(figref.area());

Example: Abstract Method

- The area method cannot compute the area of an arbitrary figure:

  ```
  double area() {
        System.out.println("Area is undefined.");
        return 0;
  }
  ```

  Instead, area should be defined abstract in Figure:

  ```
  abstract double area() ;
  ```

Example: Abstract Method

- Points to remember about abstract method:

1. Abstract method has no body.
2. Always end the declaration with a semicolon(;).
3. It must be overridden. An abstract class must be extended and in a same way abstract method must be overridden.
4. Abstract method must be in a abstract class.