Rubric

1 mark additional for following good coding practices mentioned in the mid-sem instructions throughout the 3 questions. Partial marking is applicable.

If any answer has a compilation error, they get a 0 for that answer straightaway.

Ans 1

```
public interface Rotatable {
   public void Rotate();
}
```

- 0.5 marks for declaring the interface and adding the function Rotate

```
public interface RotateAndFly extends Rotatable{
   public void fly();
}
```

- 0.5 marks for declaring the interface and adding the function fly
- 0.5 marks for extending the interface

The function declarations do not necessarily need to be void provided they have handled the same

```
public class Rotator implements Rotatable {

        @Override
        public void rotate() {
                System.out.println("Rotator rotates");
        }
}
```

- 0.5 marks for class declaration with keywords **implements Rotatable**
  0.5 marks to be deducted **if not** annotated with Override

```
public class Flyer implements RotateAndFly {

        @Override
        public void rotate() {
                System.out.println("Flyer rotates");
        }

        @Override
```

```
	public void fly() {
		System.out.println("Flyer flies");
	}
}
```

- 1 mark for class declaration with keywords **implements RotateAndFly,**
  0.5 marks to be deducted once **if not** annotated with Override in either of the functions.
Both fly() and Rotate() need to be overridden. 0 if either is missing

```
public class Helicopter implements RotateAndFly {

	@Override
	public void rotate() {
		System.out.println("Helicoptor rotates");
	}

	@Override
	public void fly() {
		System.out.println("Helicoptor flies");
	}

	public void drive(Rotatable r) {
		r.rotate();
		System.out.println(r.getClass());
	}

	public static void main(String[] args) {
		Helicopter helicoptor = new Helicopter();
		helicoptor.drive(new Rotator());
		helicoptor.drive(new Flyer());
		helicoptor.drive(new Helicopter());
	}
}
```

- 0.5 marks for implements and defining both functions(if either missing award a 0)
- 0.5 marks for declared type as Rotatable (No marks if Actual Type is the same as the declared type)
- 1 mark for the correct output for all three classes of the objects using **getClass ONLY** **(**Code execution for this 1 mark is necessary if code does not execute award a 0)

Ans 2:

```
public class Address {

	private String area;
	private String city;
	private String state;
	private int pincode;

	public Address(String area, String city, String state, int pincode) {
		this.area = area;
		this.city = city;
		this.state = state;
		this.pincode = pincode;
	}

	public String getArea() {
		return area;
	}
	public void setArea(String area) {
		this.area = area;
	}

	public String getCity() {
		return city;
	}
	public void setCity(String city) {
		this.city = city;
	}

	public int getPincode() {
		return pincode;
	}
	public void setPincode(int pincode) {
		this.pincode = pincode;
	}

	public String getState() {
		return state;
```

```
    }

    public void setState(String state) {
        this.state = state;
    }
}
```

- 1 mark for correct address class

```
public abstract class Person {

    private String firstName;
    private String lastName;
    private int id;
    private Address address;

    public Person(String firstName, String lastName, int id, Address address) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.id = id;
        this.address = address;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public int getId() {
        return id;
```

```
	}

	public void setId(int id) {
		this.id = id;
	}

	public Address getAddress() {
		return address;
	}

	public void setAddress(Address address) {
		this.address = address;
	}

	public abstract void goToWork();

}
```

- 1 mark for abstract keyword in class and in method goToWork
- 1 mark for all getters and setters

```
import java.util.Comparator;

public class DistanceComparator implements Comparator<Student> {

	@Override
	public int compare(Student o1, Student o2) {
		return -1 * (Math.abs((o1.getAddress().getPincode() - 110020))
					- Math.abs((o2.getAddress().getPincode() - 110020)));

		/* The following is equivalent.
		 * return -1 * ((o1.getAddress().getPincode())
					- (o2.getAddress().getPincode()));
		 */
	}
}
```

- 0.25 marks for using the appropriate type while implementing Comparator interface (implements Comparator<Student>).
- 0.75 marks for implementing the correct logic for the distance function

```
import java.util.Comparator;

public class CGPAComparator implements Comparator<Student> {

	@Override
	public int compare(Student o1, Student o2) {
		if (o1.getCGPA() < o2.getCGPA())
			return -1;
		else if (o1.getCGPA() > o2.getCGPA())
			return 1;
		else
			return 0;
	}

}
```

- 0.25 marks for implements Comparator<Student>
- 0.75 marks for the correct implementation.

```
import java.util.Comparator;

public class DistanceCGPAComparator implements Comparator<Student> {

	@Override
	public int compare(Student o1, Student o2) {
		double lhsDistanceCGPA = (o1.getAddress().getPincode() - 110020) -
6*o1.getCGPA();
		double rhsDistanceCGPA = (o2.getAddress().getPincode() - 110020) -
6*o2.getCGPA();

		if (lhsDistanceCGPA < rhsDistanceCGPA)
			return -1;
		else if (lhsDistanceCGPA > rhsDistanceCGPA)
			return 1;
		else
			return 0;
	}
}
```

- 0.25 marks for implements Comparator<Student>
- 0.75 marks for the correct implementation as given above.

```
import java.util.ArrayList;
import java.util.List;

public class Student extends Person {

        private double cgpa;
        private int startYear;
        private String residentialStatus;

        private static final String ON_CAMPUS = "onCampus";
        private static final String DAY_SCHOLAR = "dayScholar";

        public Student(String firstName, String lastName, int id, Address address,
                        double cgpa, int startYear, String residentialStatus) {
                super(firstName, lastName, id, address);
                this.cgpa = cgpa;
                this.startYear = startYear;
                this.residentialStatus = residentialStatus;
        }

        public double getCGPA() {
                return cgpa;
        }

        public void setCGPA(double cgpa) {
                this.cgpa = cgpa;
        }

        public int getStartYear() {
                return startYear;
        }

        public void setStartYear(int startYear) {
                this.startYear = startYear;
        }

        public String getResidentialStatus() {
                return residentialStatus;
        }

        public void setResidentialStatus(String residentialStatus) {
                this.residentialStatus = residentialStatus;
        }
```

```
    @Override
    public void goToWork() {
        if (residentialStatus == DAY_SCHOLAR)
            System.out.println("Go by vehicle");
        else if (residentialStatus == ON_CAMPUS)
            System.out.println("Go by walk");
    }

    @Override
    public String toString() {
        return "Name: " + getFirstName() + " " + getLastName() + "\n" +
                "ID: " + getId() + "\n" +
                "CGPA: " + getCGPA() + "\n" +
                "Pincode: " + getAddress().getPincode() + "\n" +
                "Residential Status: " + getResidentialStatus();
    }
```

- 0.5 marks for toString() method implementation for Student class.

```
    public static void main(String[] args) {
        List<Student> studentList = new ArrayList<Student>();

        Address address1 = new Address("Mayur Vihar", "New Delhi", "Delhi", 110024);
        Address address2 = new Address("Kalkaji", "New Delhi", "Delhi", 110015);
        Address address3 = new Address("Govindpuri", "New Delhi", "Delhi", 110022);
        Address address4 = new Address("Saket", "New Delhi", "Delhi", 110006);
        Address address5 = new Address("GK1", "New Delhi", "Delhi", 110029);

        studentList.add(new Student("f1", "l1", 123, address1, 4.5, 2021,
DAY_SCHOLAR));
        studentList.add(new Student("f2", "l2", 456, address2, 5.5, 2020,
DAY_SCHOLAR));
        studentList.add(new Student("f3", "l3", 777, address3, 6.5, 2022,
DAY_SCHOLAR));
        studentList.add(new Student("f4", "l4", 888, address4, 5.3, 2022,
DAY_SCHOLAR));
        studentList.add(new Student("f5", "l5", 999, address5, 8.5, 2021,
DAY_SCHOLAR));

        System.out.println("Distance based ranking" + "\n");
        studentList.sort(new DistanceComparator());
        for(Student s : studentList)
				System.out.println(s);

			System.out.println("\nCGPA based ranking" + "\n");
			studentList.sort(new CGPAComparator());
			for(Student s : studentList)
				System.out.println(s);

			System.out.println("\nDistance-CGPA based ranking" + "\n");
			studentList.sort(new DistanceCGPAComparator());
			for(Student s : studentList)
				System.out.println(s);

			studentList.get(0).setResidentialStatus(ON_CAMPUS);
			studentList.get(1).setResidentialStatus(ON_CAMPUS);
			studentList.get(2).setResidentialStatus(ON_CAMPUS);

			System.out.println("\nDistance-CGPA based ranking with residential status
changed" + "\n");
			for(Student s : studentList)
				System.out.println(s);
		}
}
```

- 0.5 mark for making atleast 5 students
- 0.25 marks for correct printing order of CGPA(lowest first)
- 0.25 marks for correct printing order of distance(highest first)
- 0.5 mark for correct printing order of goToWork based on the defined criteria

Ans 3:

- 1 mark for creating a separate class and extending Exception
- 1 mark for constructor

```
public class IncorrectPincodeException extends Exception {

	private static final long serialVersionUID = 1L;

	public IncorrectPincodeException(String message) {
		super(message);
	}

}
```

- 1 mark for using try-catch block
- 0.5 marks for correctly retaking input
- 2.5 marks for correctly throwing the error at each condition.
  Deduct half mark for each condition missed.
- 0.5 marks for explicitly catching the IncorrectPincodeException error
- 0.5 mark for printing the error message

```
import java.util.Scanner;

public class PincodeVerifier {

	public boolean verifyPincode(String pincode) throws IncorrectPincodeException {
		boolean pincodeFlag = true;

		if (pincode == null) {
			pincodeFlag = false;
			throw new IncorrectPincodeException("pincode is null");
		}
		else if (pincode.length() == 0) {
			pincodeFlag = false;
			throw new IncorrectPincodeException("pincode is empty");
		}
		else if (pincode.length() < 6) {
			pincodeFlag = false;
			throw new IncorrectPincodeException("pincode length is less than 6");
		}
		else if (pincode.length() > 6) {
			pincodeFlag = false;
			throw new IncorrectPincodeException("pincode length is more than 6");
		}
		else if (pincode.charAt(0) == '0') {
			pincodeFlag = false;
			throw new IncorrectPincodeException("pincode begins with a 0");
		}

		for(int i = 0; i < pincode.length(); i++) {
			if (pincode.charAt(i) >= '0' && pincode.charAt(i) <= '9')
				pincodeFlag = true;
			else {
				pincodeFlag = false;
				throw new IncorrectPincodeException("pincode does not have
numeric digits");
			}
		}
		return pincodeFlag;
	}

	public static void main(String[] args) {
		String pincode;
		Scanner scanner = new Scanner(System.in);
		PincodeVerifier pincodeVerifier = new PincodeVerifier();
		boolean pincodeFlag = false;

		while(!pincodeFlag) {
			System.out.println("\nEnter the pincode: ");
			pincode = scanner.nextLine();
			try {
				pincodeFlag = pincodeVerifier.verifyPincode(pincode);
			} catch (IncorrectPincodeException e) {
//				System.out.println(e.getMessage());
				e.printStackTrace();
				System.out.println();
			}
		}
		System.out.println("Done");
	}
}
```