```csharp
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using System;
using System.Collections.Generic;
using System.Text;

namespace GameDemo.Shared
{
    public enum ENEMY_TYPES { SKELETON = 1, BOULDER = 2, AXE = 3, GHOST = 4 }

    public class Enemy :Sprite
    {

        bool isVisible;
        public float startingYpos;

        ENEMY_TYPES type;


        // shows whether the player is visible
        public bool Visibility
        { get { return isVisible; } }


        public ENEMY_TYPES Type
        {
            get { return (ENEMY_TYPES) type; }
            set { type = value; }
        }

        public Vector2 Speed
        {
            get { return speed; }
        }

        public Enemy(ENEMY_TYPES enemyType,Texture2D texture, Vector2 position, Point frameSize,
            int totalFrames, Vector2 collisionOffset, Point currentFrame, Point sheetSize,
            Vector2 speed,int secondsperframe,float scale)

            : base(texture, position, frameSize, totalFrames, collisionOffset, currentFrame, sheetSize, speed,scale)
        {
            millisecondsPerFrame = secondsperframe;
            startingYpos = position.Y;
            type = enemyType;
        }

        // get the collision rectangle
        public override Rectangle collisionRect()
        {
            if(this.type==ENEMY_TYPES.BOULDER)
                  return new Rectangle((int)position.X-frameSize.X, (int)position.Y-frameSize.Y, (int)((float)this.frameSize.X * scale - collisionOffset.X), (int)((float)this.frameSize.Y
                         * scale - this.collisionOffset.Y));
            else
                return new Rectangle((int)position.X, (int)position.Y, (int)
                  ((float)this.frameSize.X*scale-collisionOffset.X),(int)
                  ((float)this.frameSize.Y*scale- this.collisionOffset.Y));
        }

        // where the enemies spawn
        public virtual TimeSpan Spawn(GameTime gameTime)
        {
            switch (type)
            {
                case ENEMY_TYPES.AXE:
                    {
                        isVisible = true;
                        position.X = 1900;
                        //position.Y = 800;
                        break;
                    }
                case ENEMY_TYPES.SKELETON:
                    {
                        isVisible = true;
                        position.X = 1900;
                        //position.Y = 800;
                        break;
                    }
                case ENEMY_TYPES.BOULDER:
                    {
                        isVisible = true;
                        position.X = 1900;

                        position.Y = 400;
                        break;
                    }
                case ENEMY_TYPES.GHOST:
                    {
                        isVisible = true;
                        position.X = 1900;
                        position.Y = 500;
                        break;
                    }
            }
            return gameTime.TotalGameTime;
        }
        public void Behaviour(GameTime gameTime, ref float timeOnScreen) {

            switch (type)
            {
                case ENEMY_TYPES.AXE:
                    {
                        position.X -= Speed.X;
                        if (position.X < -200 || position.X > 1920)
                            isVisible = false;
                        break;
                    }
                case ENEMY_TYPES.SKELETON:
                    {
                        position.X -= Speed.X;
                        if (position.X < -200 || position.X > 1920)
                            isVisible = false;
                        break;
                    }
                case ENEMY_TYPES.BOULDER:
                    {
                        position.X -= Speed.X;
                        if (position.X < 0 || position.X > 1920)
                            isVisible = false;
                        position = Movement.Bouncing(position,3, gameTime,ref timeOnScreen);
                        break;
                    }
                case ENEMY_TYPES.GHOST:
                    {
                        position.X -= Speed.X;
                        if (position.X < 0 || position.X > 1920)
                            isVisible = false;
                        position = Movement.SinWave(position, 20, 0.2, gameTime,ref timeOnScreen);
                        break;
                    }
            }
        }

        public override void Draw(GameTime gametime, SpriteBatch spritebatch, float scale, SpriteEffects spriteEffects)
        {
            if(isVisible)
            base.Draw(gametime, spritebatch, scale, spriteEffects);

        }

        public void DrawRotating(GameTime gametime, SpriteBatch spritebatch, float scale, SpriteEffects spriteEffects,float rotation)
        {
            if(isVisible)
            spritebatch.Draw(Texture, position, new Rectangle(currentFrame.X * frameSize.X, currentFrame.Y * frameSize.Y, frameSize.X, frameSize.Y), Color.White, rotation, new Vector2(frameSize.X/2,frameSize.Y/2), scale, spriteEffects, 0);
        }
    }
}
```